Education of Embedded Systems Programming in C and Assembly Based on ARM's Cortex-M Microprocessors

ARM

life.augmented

Yifeng Zhu

University of Maine

ASEE Annual Conference
June 2016

Install Software and Driver

- Each of you should have
  - STM32L4 Discovery Kit
  - USB Cable
  - USB Flash Drive

- Let's get the installation started before the presentation
  - Step 1: Insert the USB flash to the laptop
  - Step 2: Install Keil µVision. *Keil can only run Windows or Windows VM!*
    - If you have a Windows laptop, please run "MDK520.EXE" to install Keil µVision v5.20
    - If you have a Mac or Linux, please (1) Download VirtualBox from www.virtualbox.org and install it, and (2) Import the Window image from the USB flash drive

Do not plug in the STM32L4 discovery kit into your laptop until instructed to do so.

Experiential Learning

“I hear and I forget. I see and I remember. I do and I understand.”

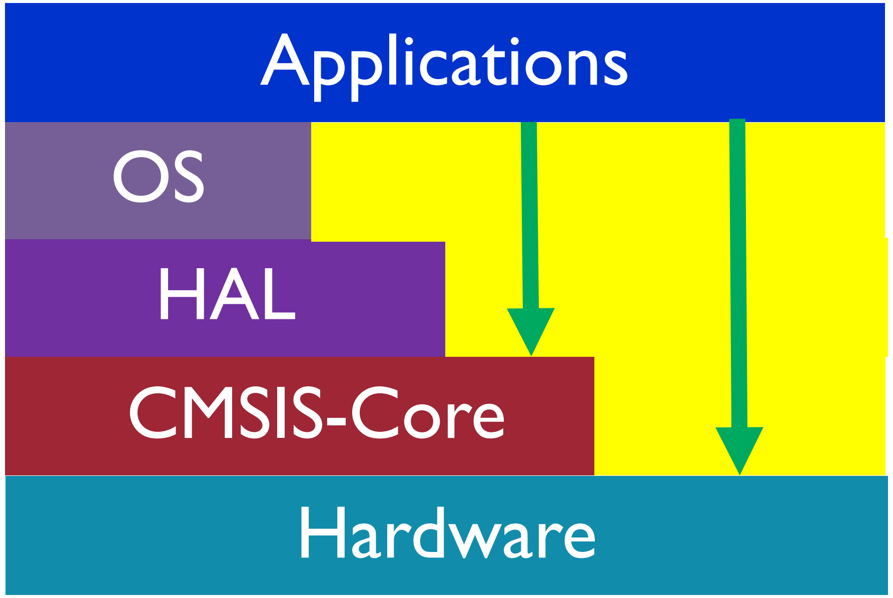
-- Confucius (Chinese philosopher, 551–479 BC)

“Tell me and I forget, teach me and I remember, involve me and I will learn.”

-- Benjamin Franklin, 1706-1790

Teach at Which Level?

- Visual wizard tools (such as STMCubeMX)
- HAL (Hardware Abstraction Layer) libraries
- Bare-metal



Bare-metal: Bypass OS, HAL and possibly CMSIS-Core

HAL Level

```
; Initialize the Red LED pin (PB.2)
static GPIO_InitTypeDef  GPIO_InitStruct;
GPIO_InitStruct.Mode  = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull  = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Pin   = GPIO_PIN_2;

HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

HAL_GPIO_TogglePin(LED4_GPIO_PORT, LED4_PIN);
```

- Pros
  - Simplify implementation
  - Better portability
  - Many examples
- Cons
  - Very complex to understand
  - Cannot meet students' curiosity
  - Does not facilitate deep learning

```
void HAL_GPIO_Init(GPIO_TypeDef  *GPIOx, GPIO_InitTypeDef *GPIO_Init){
  uint32_t position = 0x00;
  uint32_t iocurrent = 0x00;
  uint32_t temp = 0x00;
  ...
}
```

130 lines

Learning to see the forest and the trees!

Bare-Metal Level in C

```
#define LED_PIN 2

// GPIO Mode: Input(00), Output(01), AlterFunc(10), Analog(11, reset)
GPIOB->MODER &= ~(3<<(2*LED_PIN));
GPIOB->MODER |=   1<<(2*LED_PIN);     // Output(01)

// GPIO Speed: Low speed (00), Medium speed (01), Fast speed (10), High speed (11)
GPIOB->OSPEEDR &= ~(3<<(2*LED_PIN));
GPIOB->OSPEEDR |=   2<<(2*LED_PIN);   // Fast speed

// GPIO Output Type: Output push-pull (0, reset), Output open drain (1)
GPIOB->OTYPER &= ~(1<<LED_PIN);       // Push-pull

// GPIO Push-Pull: No pull-up pull-down (00), Pull-up (01), Pull-down (10),
Reserved (11)
GPIOB->PUPDR &= ~(3<<(2*LED_PIN));  // No pull-up, no pull-down

// Toggle up the LED
GPIOB->ODR ^= 1 << LED_PIN;
```

- Focus on directly interfacing with hardware.
- Do not use any libraries!

Bare-Metal Level in Assembly

Bare-metal level programming helps learning assembly programming

Set Pin B.2 as GPIO output

```
#define LED_PIN 2

// GPIO Mode: Input(00), Output(01), AlterFunc(10), Analog(11, reset)
GPIOB->MODER &= ~(3<<(2*LED_PIN));
GPIOB->MODER |=   1<<(2*LED_PIN);    // Output(01)
```

C implementation

Translate naturally

```
LED_PIN EQU 2

   LDR r0, =GPIOB_BASE
   LDR r1, [r0, #GPIO_MODER]
   EOR r1, r1, #(0x03<<(2*LED_PIN))
   ORR r1, r1, #(1<<LED_PIN)
   STR r1, [r0, #GPIO_MODER]
```

Assembly implementation

Why assembly?

- Help write efficient programs in high-level languages
- Best for performance-critical or latency-sensitive applications
- Understanding hardware–software interactions
- Basic ingredients for computer architecture and operating systems courses

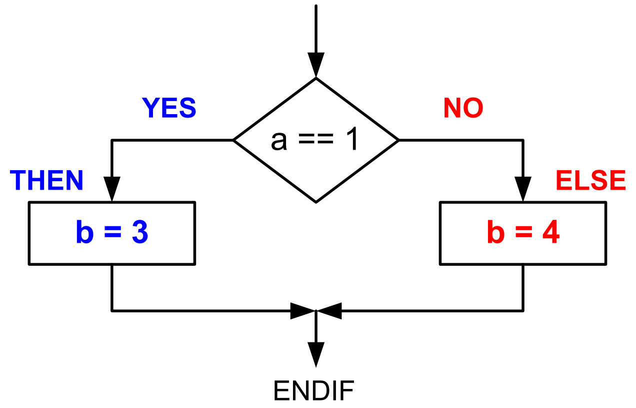
A Structured Approach in Assembly Programming

- Assembly is not a structured programming language
  - No high-level control constructs to avoid GOTOs (unconditional branches)
  - Difficulty to learn and program
  - Prone to create spaghetti codes

A Structured Approach in Assembly Programming

Methods of teaching structured programming in assembly

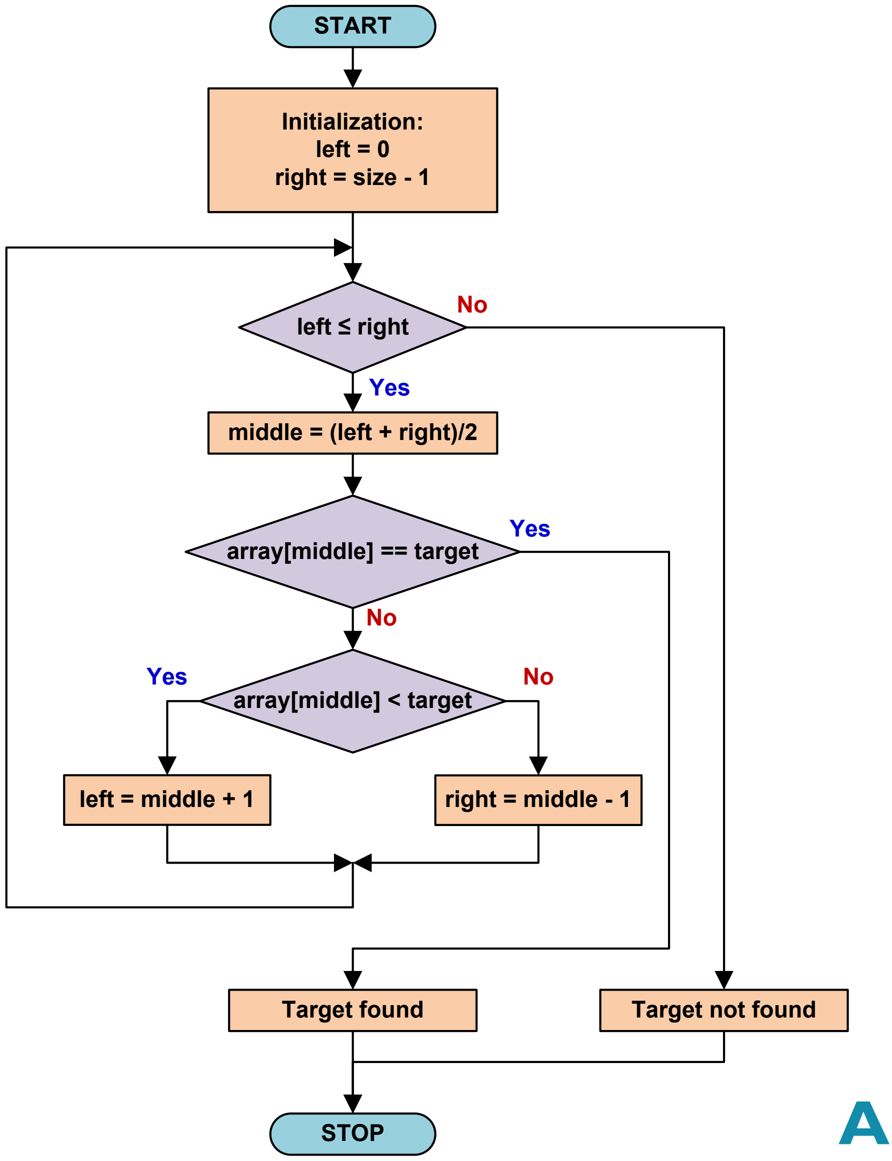
- Use of flowcharts
  - Separate program structuring from code writing



A Structured Approach in Assembly Programming

Methods of teaching structured programming in assembly

- Use of flowcharts
  - Separate program structuring from code writing

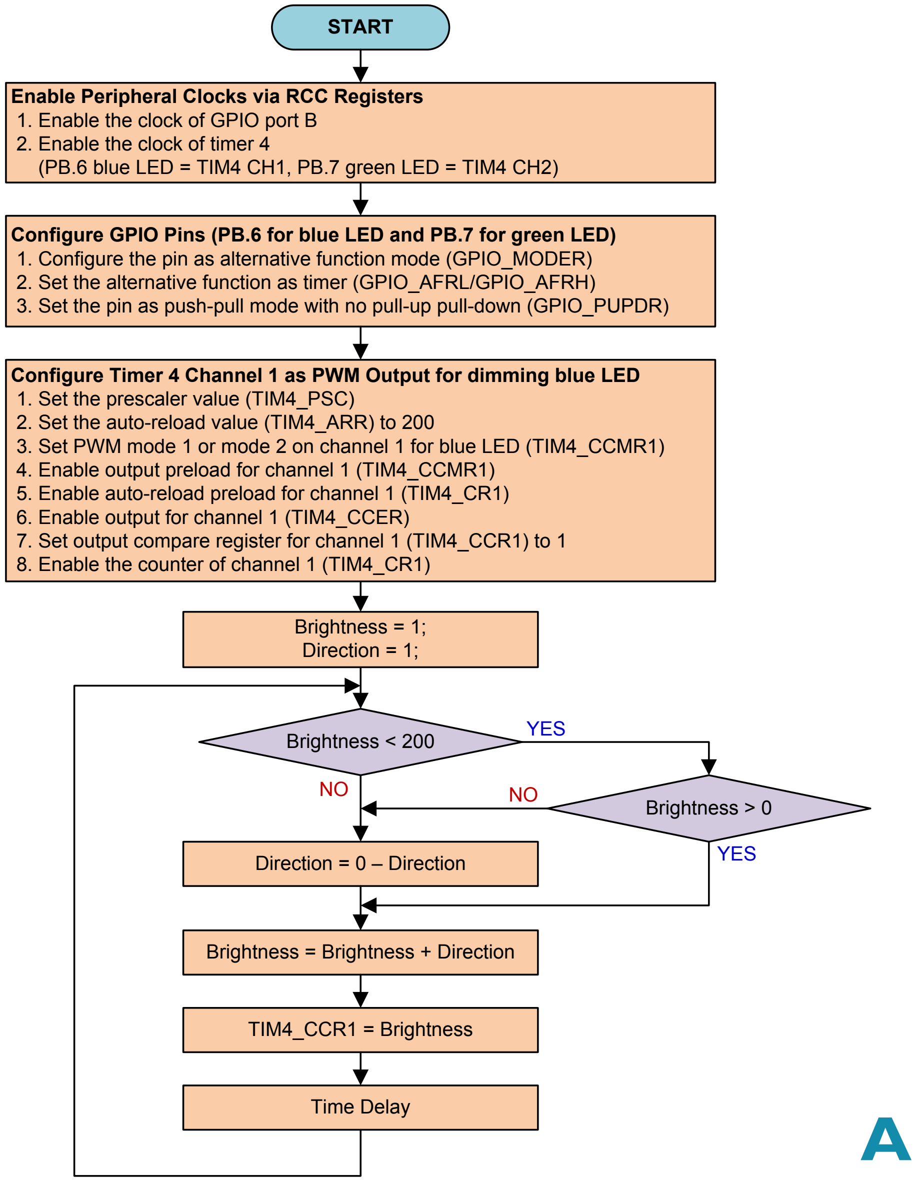


Flowchart of Binary Search

A Structured Approach in Assembly Programming

Methods of teaching structured programming in assembly

- Use of flowcharts
  - Separate program structuring from code writing



START

Enable Peripheral Clocks via RCC Registers
1. Enable the clock of GPIO port B
2. Enable the clock of timer 4
   (PB.6 blue LED = TIM4 CH1, PB.7 green LED = TIM4 CH2)

Configure GPIO Pins (PB.6 for blue LED and PB.7 for green LED)
1. Configure the pin as alternative function mode (GPIO_MODER)
2. Set the alternative function as timer (GPIO_AFRL/GPIO_AFRH)
3. Set the pin as push-pull mode with no pull-up pull-down (GPIO_PUPDR)

Configure Timer 4 Channel 1 as PWM Output for dimming blue LED
1. Set the prescaler value (TIM4_PSC)
2. Set the auto-reload value (TIM4_ARR) to 200
3. Set PWM mode 1 or mode 2 on channel 1 for blue LED (TIM4_CCMR1)
4. Enable output preload for channel 1 (TIM4_CCMR1)
5. Enable auto-reload preload for channel 1 (TIM4_CR1)
6. Enable output for channel 1 (TIM4_CCER)
7. Set output compare register for channel 1 (TIM4_CCR1) to 1
8. Enable the counter of channel 1 (TIM4_CR1)

Brightness = 1;
Direction = 1;

Brightness < 200

Brightness > 0

Direction = 0 – Direction

Brightness = Brightness + Direction

TIM4_CCR1 = Brightness

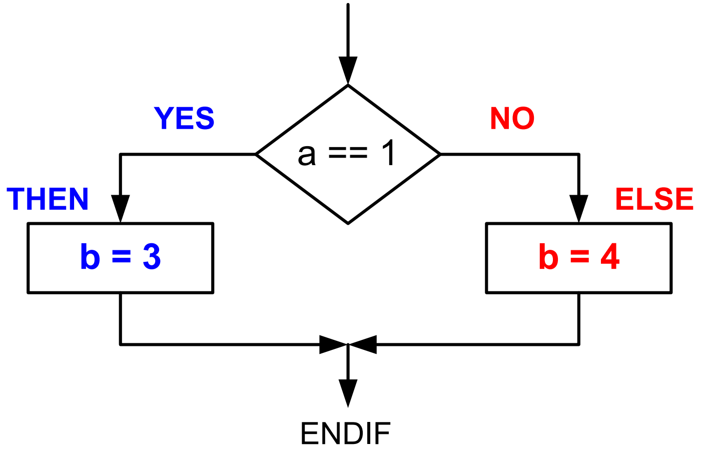
Time Delay

Diming LED by using timer PWM output

A Structured Approach in Assembly Programming

Methods of teaching structured programming in assembly

- Use of flowcharts
  - Separate program structuring from code writing
- C-Assembly line-by-line comparison
  - Relate an unstructured to a structured

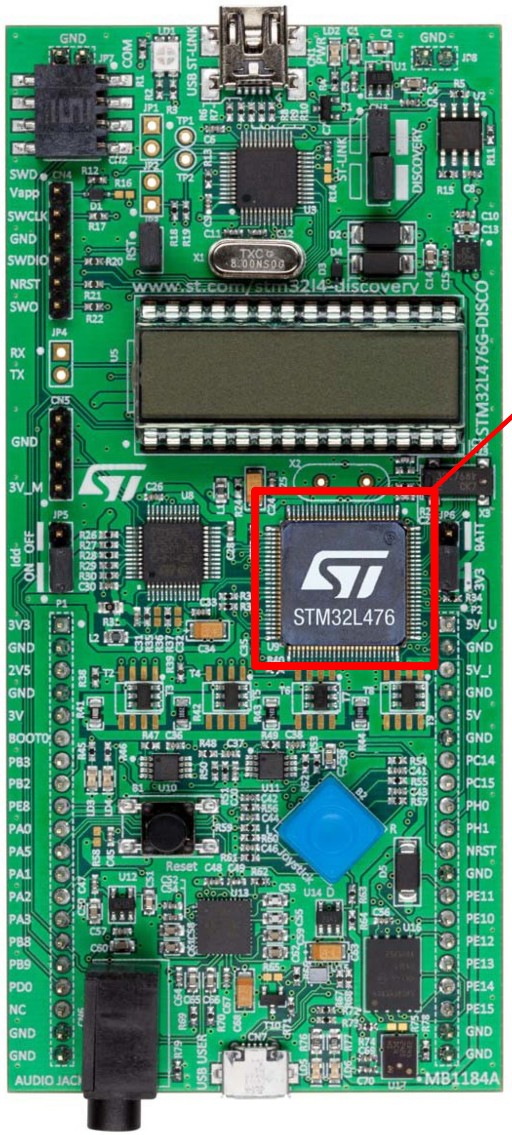


C Program	Assembly Program

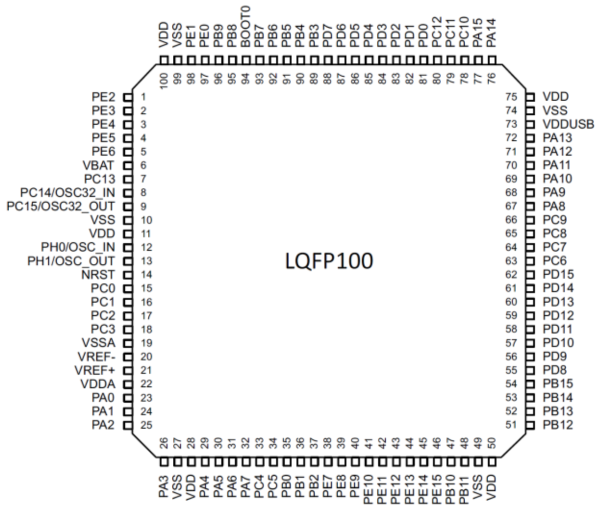
if (a == 1)
  b = 3
else
  b = 4;
	; r1 = a, r2 = b
	     CMP r1, #1
	     BNE else
	then MOV r2, #3
	     B endif
	else MOV r2, #4
	endif

Selecting a Platform: Hardware Component

- Low cost
  - ~$20 each
- Hands-on experiences
  - develop and test real systems
- Rewarding and engaging
  - immediately enjoy the fruit of labor
- Convenient
  - mobile lab without time and location constrains
- Versatile
  - pins are extended for easy access



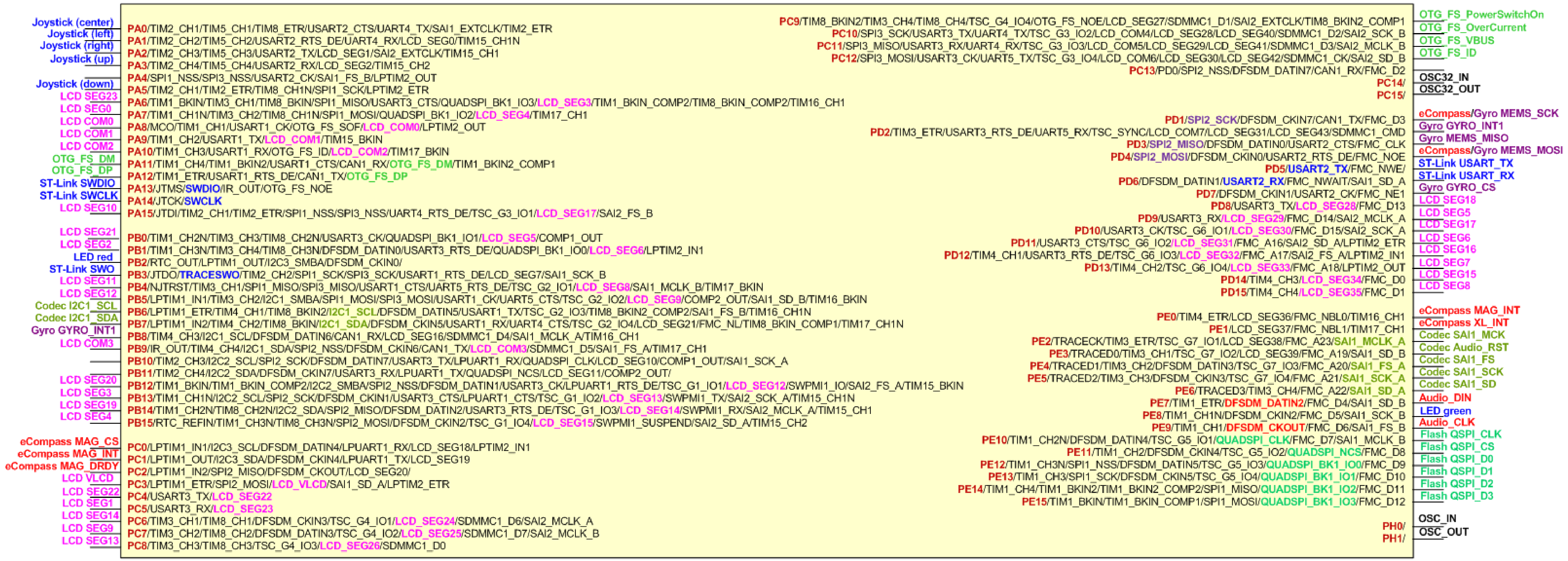
STM32L476G



STM32L4 Discovery Kit @STMicroelectronics

STM32L476G

Joystick (MT-008A): Center = PA0, Left = PA1, Right = PA2, Up = PA3, Down = PA5
User LED: Red = PB2, Green = PE8
Analog Outputs: PA3 (OPAMP1_OUT), PA5 (DAC1_OUT2). *Note: PA4 (DAC1_OUT1) is not extended out, thus we use OPAMP1_OUT.*



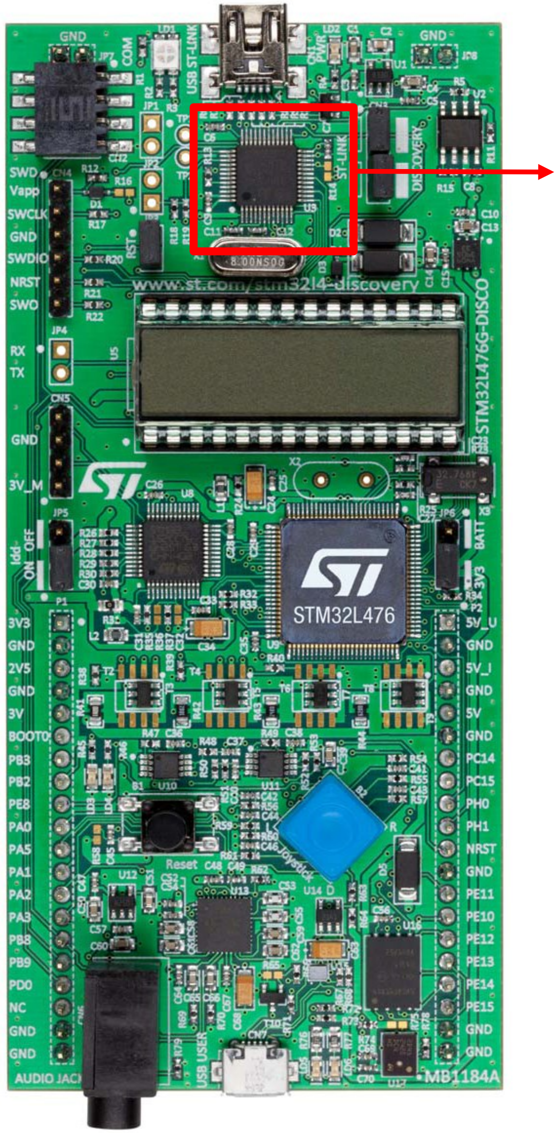
Joystick (center)	PA0/TIM2_CH1/TIM5_CH1/TIM8_ETR/USART2_CTS/UART4_TX/SAI1_EXTCLK/TIM2_ETR
Joystick (left)	PA1/TIM2_CH2/TIM5_CH2/USART2_RTS_DE/UART4_RX/LCD_SEG0/TIM15_CH1N
Joystick (right)	PA2/TIM2_CH3/TIM5_CH3/USART2_TX/LCD_SEG1/SAI2_EXTCLK/TIM15_CH1
Joystick (up)	PA3/TIM2_CH4/TIM5_CH4/USART2_RX/LCD_SEG2/TIM15_CH2
	PA4/SPI1_NSS/SPI3_NSS/USART2_CK/SAI1_FS_B/LPTIM2_OUT
Joystick (down)	PA5/TIM2_CH1/TIM2_ETR/TIM8_CH1N/SPI1_SCK/LPTIM2_ETR
LCD SEG23	PA6/TIM1_BKIN/TIM3_CH1/TIM8_BKIN/SPI1_MISO/USART3_CTS/QUADSPI_BK1_IO3/LCD_SEG3/TIM1_BKIN_COMP2/TIM8_BKIN_COMP2/TIM16_CH1
LCD SEG0	PA7/TIM1_CH1N/TIM3_CH2/TIM8_CH1N/SPI1_MOSI/QUADSPI_BK1_IO2/LCD_SEG4/TIM17_CH1
LCD COM0	PA8/MCO/TIM1_CH1/USART1_CK/OTG_FS_SOF/LCD_COM0/LPTIM2_OUT
LCD COM1	PA9/TIM1_CH2/USART1_TX/LCD_COM1/TIM15_BKIN
LCD COM2	PA10/TIM1_CH3/USART1_RX/OTG_FS_ID/LCD_COM2/TIM17_BKIN
OTG_FS_DM	PA11/TIM1_CH4/TIM1_BKIN2/USART1_CTS/CAN1_RX/OTG_FS_DM/TIM1_BKIN2_COMP1
OTG_FS_DP	PA12/TIM1_ETR/USART1_RTS_DE/CAN1_TX/OTG_FS_DP
ST-Link SWDIO	PA13/JTMS/SWDIO/IR_OUT/OTG_FS_NOE
ST-Link SWCLK	PA14/JTCK/SWCLK
LCD SEG10	PA15/JTDI/TIM2_CH1/TIM2_ETR/SPI1_NSS/SPI3_NSS/UART4_RTS_DE/TSC_G3_IO1/LCD_SEG17/SAI2_FS_B
LCD SEG21	PB0/TIM1_CH2N/TIM3_CH3/TIM8_CH2N/USART3_CK/QUADSPI_BK1_IO1/LCD_SEG5/COMP1_OUT
LCD SEG2	PB1/TIM1_CH3N/TIM3_CH4/TIM8_CH3N/DFSDM_DATIN0/USART3_RTS_DE/QUADSPI_BK1_IO0/LCD_SEG6/LPTIM2_IN1
LED red	PB2/RTC_OUT/LPTIM1_OUT/I2C3_SMBA/DFSDM_CKIN0/
ST-Link SWO	PB3/JTDO/TRACESWO/TIM2_CH2/SPI1_SCK/SPI3_SCK/USART1_RTS_DE/LCD_SEG7/SAI1_SCK_B
LCD SEG11	PB4/NJTRST/TIM3_CH1/SPI1_MISO/SPI3_MISO/USART1_CTS/UART5_RTS_DE/TSC_G2_IO1/LCD_SEG8/SAI1_MCLK_B/TIM17_BKIN
LCD SEG12	PB5/LPTIM1_IN1/TIM3_CH2/I2C1_SMBA/SPI1_MOSI/SPI3_MOSI/USART1_CK/UART5_CTS/TSC_G2_IO2/LCD_SEG9/COMP2_OUT/SAI1_SD_B/TIM16_BKIN
Codec I2C1_SCL	PB6/LPTIM1_ETR/TIM4_CH1/TIM8_BKIN2/I2C1_SCL/DFSDM_DATIN5/USART1_TX/TSC_G2_IO3/TIM8_BKIN2_COMP2/SAI1_FS_B/TIM16_CH1N
Codec I2C1_SDA	PB7/LPTIM1_IN2/TIM4_CH2/TIM8_BKIN/I2C1_SDA/DFSDM_CKIN5/USART1_RX/UART4_CTS/TSC_G2_IO4/LCD_SEG21/FMC_NL/TIM8_BKIN_COMP1/TIM17_CH1N
Gyro GYRO_INT1	PB8/TIM4_CH3/I2C1_SCL/DFSDM_DATIN6/CAN1_RX/LCD_SEG16/SDMMC1_D4/SAI1_MCLK_A/TIM16_CH1
LCD COM3	PB9/IR_OUT/TIM4_CH4/I2C1_SDA/SPI2_NSS/DFSDM_CKIN6/CAN1_TX/LCD_COM3/SDMMC1_D5/SAI1_FS_A/TIM17_CH1
	PB10/TIM2_CH3/I2C2_SCL/SPI2_SCK/DFSDM_DATIN7/USART3_TX/LPUART1_RX/QUADSPI_CLK/LCD_SEG10/COMP1_OUT/SAI1_SCK_A
	PB11/TIM2_CH4/I2C2_SDA/DFSDM_CKIN7/USART3_RX/LPUART1_TX/QUADSPI_NCS/LCD_SEG11/COMP2_OUT/
LCD SEG20	PB12/TIM1_BKIN/TIM1_BKIN_COMP2/I2C2_SMBA/SPI2_NSS/DFSDM_DATIN1/USART3_CK/LPUART1_RTS_DE/TSC_G1_IO1/LCD_SEG12/SWPMI1_IO/SAI2_FS_A/TIM15_BKIN
LCD SEG3	PB13/TIM1_CH1N/I2C2_SCL/SPI2_SCK/DFSDM_CKIN1/USART3_CTS/LPUART1_CTS/TSC_G1_IO2/LCD_SEG13/SWPMI1_TX/SAI2_SCK_A/TIM15_CH1N
LCD SEG19	PB14/TIM1_CH2N/TIM8_CH2N/I2C2_SDA/SPI2_MISO/DFSDM_DATIN2/USART3_RTS_DE/TSC_G1_IO3/LCD_SEG14/SWPMI1_RX/SAI2_MCLK_A/TIM15_CH1
LCD SEG4	PB15/RTC_REFIN/TIM1_CH3N/TIM8_CH3N/SPI2_MOSI/DFSDM_CKIN2/TSC_G1_IO4/LCD_SEG15/SWPMI1_SUSPEND/SAI2_SD_A/TIM15_CH2
eCompass MAG_CS	PC0/LPTIM1_IN1/I2C3_SCL/DFSDM_DATIN4/LPUART1_RX/LCD_SEG18/LPTIM2_IN1
eCompass MAG_INT	PC1/LPTIM1_OUT/I2C3_SDA/DFSDM_CKIN4/LPUART1_TX/LCD_SEG19
eCompass MAG_DRDY	PC2/LPTIM1_IN2/SPI2_MISO/DFSDM_CKOUT/LCD_SEG20/
LCD VLCD	PC3/LPTIM1_ETR/SPI2_MOSI/LCD_VLCD/SAI1_SD_A/LPTIM2_ETR
LCD SEG22	PC4/USART3_TX/LCD_SEG22
LCD SEG1	PC5/USART3_RX/LCD_SEG23
LCD SEG14	PC6/TIM3_CH1/TIM8_CH1/DFSDM_CKIN3/TSC_G4_IO1/LCD_SEG24/SDMMC1_D6/SAI2_MCLK_A
LCD SEG9	PC7/TIM3_CH2/TIM8_CH2/DFSDM_DATIN3/TSC_G4_IO2/LCD_SEG25/SDMMC1_D7/SAI2_MCLK_B
LCD SEG13	PC8/TIM3_CH3/TIM8_CH3/TSC_G4_IO3/LCD_SEG26/SDMMC1_D0

PC9/TIM8_BKIN2/TIM3_CH4/TIM8_CH4/TSC_G4_IO4/OTG_FS_NOE/LCD_SEG27/SDMMC1_D1/SAI2_EXTCLK/TIM8_BKIN2_COMP1	OTG_FS_PowerSwitchOn
PC10/SPI3_SCK/USART3_TX/UART4_TX/TSC_G3_IO2/LCD_COM4/LCD_SEG28/LCD_SEG40/SDMMC1_D2/SAI2_SCK_B	OTG_FS_OverCurrent
PC11/SPI3_MISO/USART3_RX/UART4_RX/TSC_G3_IO3/LCD_COM5/LCD_SEG29/LCD_SEG41/SDMMC1_D3/SAI2_MCLK_B	OTG_FS_VBUS
PC12/SPI3_MOSI/USART3_CK/UART5_TX/TSC_G3_IO4/LCD_COM6/LCD_SEG30/LCD_SEG42/SDMMC1_CK/SAI2_SD_B	OTG_FS_ID
PC13/PD0/SPI2_NSS/DFSDM_DATIN7/CAN1_RX/FMC_D2	
PC14/	OSC32_IN
PC15/	OSC32_OUT
PD1/SPI2_SCK/DFSDM_CKIN7/CAN1_TX/FMC_D3	eCompass/Gyro MEMS_SCK
PD2/TIM3_ETR/USART3_RTS_DE/UART5_RX/TSC_SYNC/LCD_COM7/LCD_SEG31/LCD_SEG43/SDMMC1_CMD	Gyro GYRO_INT1
PD3/SPI2_MISO/DFSDM_DATIN0/USART2_CTS/FMC_CLK	Gyro MEMS_MISO
PD4/SPI2_MOSI/DFSDM_CKIN0/USART2_RTS_DE/FMC_NOE	eCompass/Gyro MEMS_MOSI
PD5/USART2_TX/FMC_NWE/	ST-Link USART_TX
PD6/DFSDM_DATIN1/USART2_RX/FMC_NWAIT/SAI1_SD_A	ST-Link USART_RX
PD7/DFSDM_CKIN1/USART2_CK/FMC_NE1	Gyro GYRO_CS
PD8/USART3_TX/LCD_SEG28/FMC_D13	LCD SEG18
PD9/USART3_RX/LCD_SEG29/FMC_D14/SAI2_MCLK_A	LCD SEG5
PD10/USART3_CK/TSC_G6_IO1/LCD_SEG30/FMC_D15/SAI2_SCK_A	LCD SEG17
PD11/USART3_CTS/TSC_G6_IO2/LCD_SEG31/FMC_A16/SAI2_SD_A/LPTIM2_ETR	LCD SEG6
PD12/TIM4_CH1/USART3_RTS_DE/TSC_G6_IO3/LCD_SEG32/FMC_A17/SAI2_FS_A/LPTIM2_IN1	LCD SEG16
PD13/TIM4_CH2/TSC_G6_IO4/LCD_SEG33/FMC_A18/LPTIM2_OUT	LCD SEG7
PD14/TIM4_CH3/LCD_SEG34/FMC_D0	LCD SEG15
PD15/TIM4_CH4/LCD_SEG35/FMC_D1	LCD SEG8
PE0/TIM4_ETR/LCD_SEG36/FMC_NBL0/TIM16_CH1	eCompass MAG_INT
PE1/LCD_SEG37/FMC_NBL1/TIM17_CH1	eCompass XL_INT
PE2/TRACECK/TIM3_ETR/TSC_G7_IO1/LCD_SEG38/FMC_A23/SAI1_MCLK_A	Codec SAI1_MCK
PE3/TRACED0/TIM3_CH1/TSC_G7_IO2/LCD_SEG39/FMC_A19/SAI1_SD_B	Codec Audio_RST
PE4/TRACED1/TIM3_CH2/DFSDM_DATIN3/TSC_G7_IO3/FMC_A20/SAI1_FS_A	Codec SAI1_FS
PE5/TRACED2/TIM3_CH3/DFSDM_CKIN3/TSC_G7_IO4/FMC_A21/SAI1_SCK_A	Codec SAI1_SCK
PE6/TRACED3/TIM3_CH4/FMC_A22/SAI1_SD_A	Codec SAI1_SD
PE7/TIM1_ETR/DFSDM_DATIN2/FMC_D4/SAI1_SD_B	Audio_DIN
PE8/TIM1_CH1N/DFSDM_CKIN2/FMC_D5/SAI1_SCK_B	LED green
PE9/TIM1_CH1/DFSDM_CKOUT/FMC_D6/SAI1_FS_B	Audio_CLK
PE10/TIM1_CH2N/DFSDM_DATIN4/TSC_G5_IO1/QUADSPI_CLK/FMC_D7/SAI1_MCLK_B	Flash QSPI_CLK
PE11/TIM1_CH2/DFSDM_CKIN4/TSC_G5_IO2/QUADSPI_NCS/FMC_D8	Flash QSPI_CS
PE12/TIM1_CH3N/SPI1_NSS/DFSDM_DATIN5/TSC_G5_IO3/QUADSPI_BK1_IO0/FMC_D9	Flash QSPI_D0
PE13/TIM1_CH3/SPI1_SCK/DFSDM_CKIN5/TSC_G5_IO4/QUADSPI_BK1_IO1/FMC_D10	Flash QSPI_D1
PE14/TIM1_CH4/TIM1_BKIN2/TIM1_BKIN2_COMP2/SPI1_MISO/QUADSPI_BK1_IO2/FMC_D11	Flash QSPI_D2
PE15/TIM1_BKIN/TIM1_BKIN_COMP1/SPI1_MOSI/QUADSPI_BK1_IO3/FMC_D12	Flash QSPI_D3
PH0/	OSC_IN
PH1/	OSC_OUT

STM32L476VG Discovery Kit Pin Connection

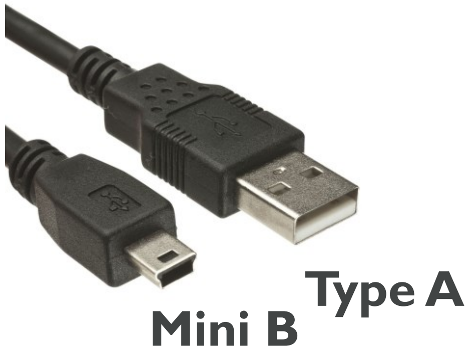
Selecting a Platform: Hardware Component

- Low cost
  - ~$20 each
- Hands-on experiences
  - develop and test real systems
- Rewarding and engaging
  - immediately enjoy the fruit of labor
- Convenient
  - mobile lab without time and location constrains
- Versatile
  - pins are extended for easy access



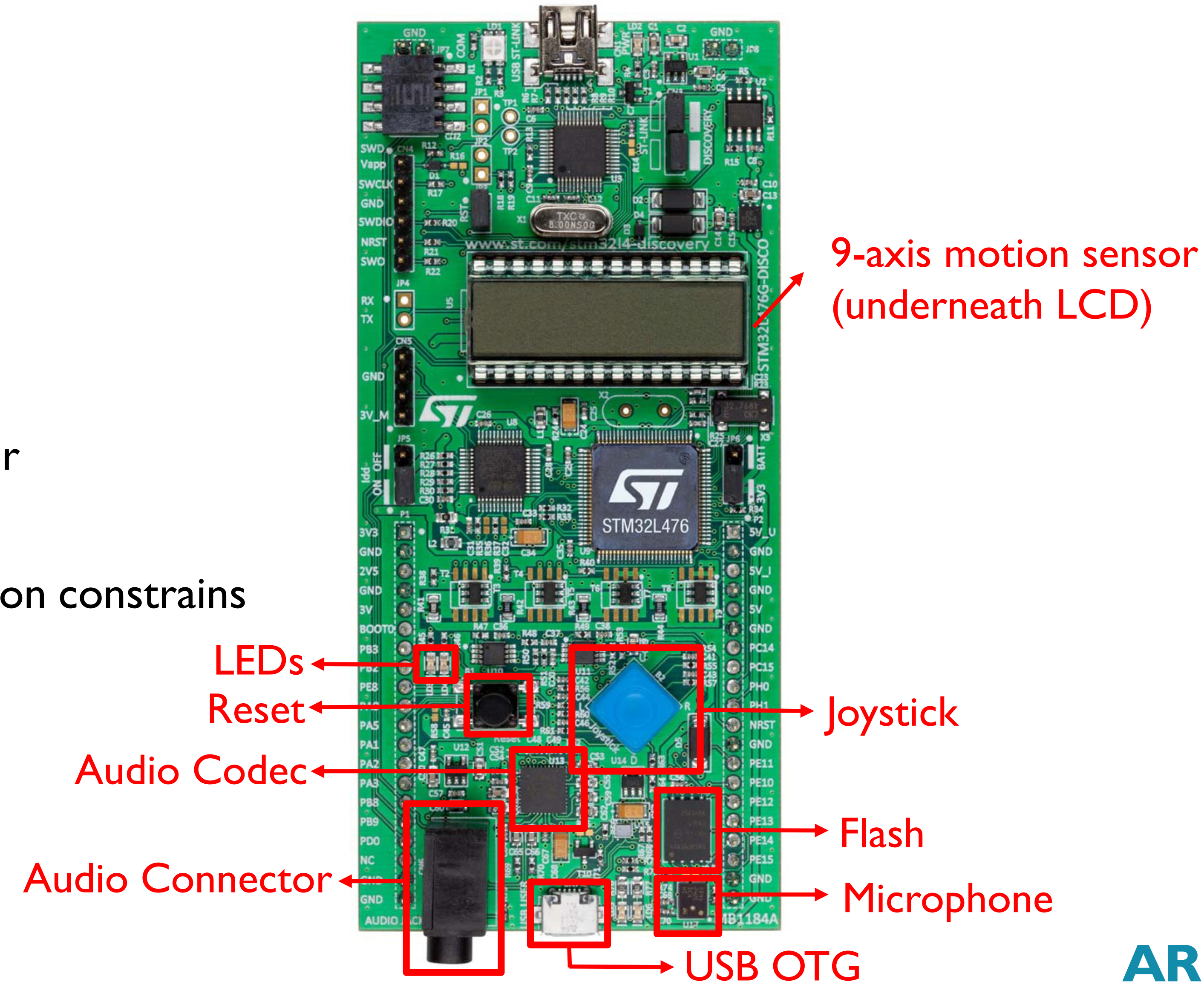
Integrated ST-Link/V2 programming and debugging tool

STM32L4 Discovery Kit @STMicroelectronics



Selecting a Platform: Hardware Component

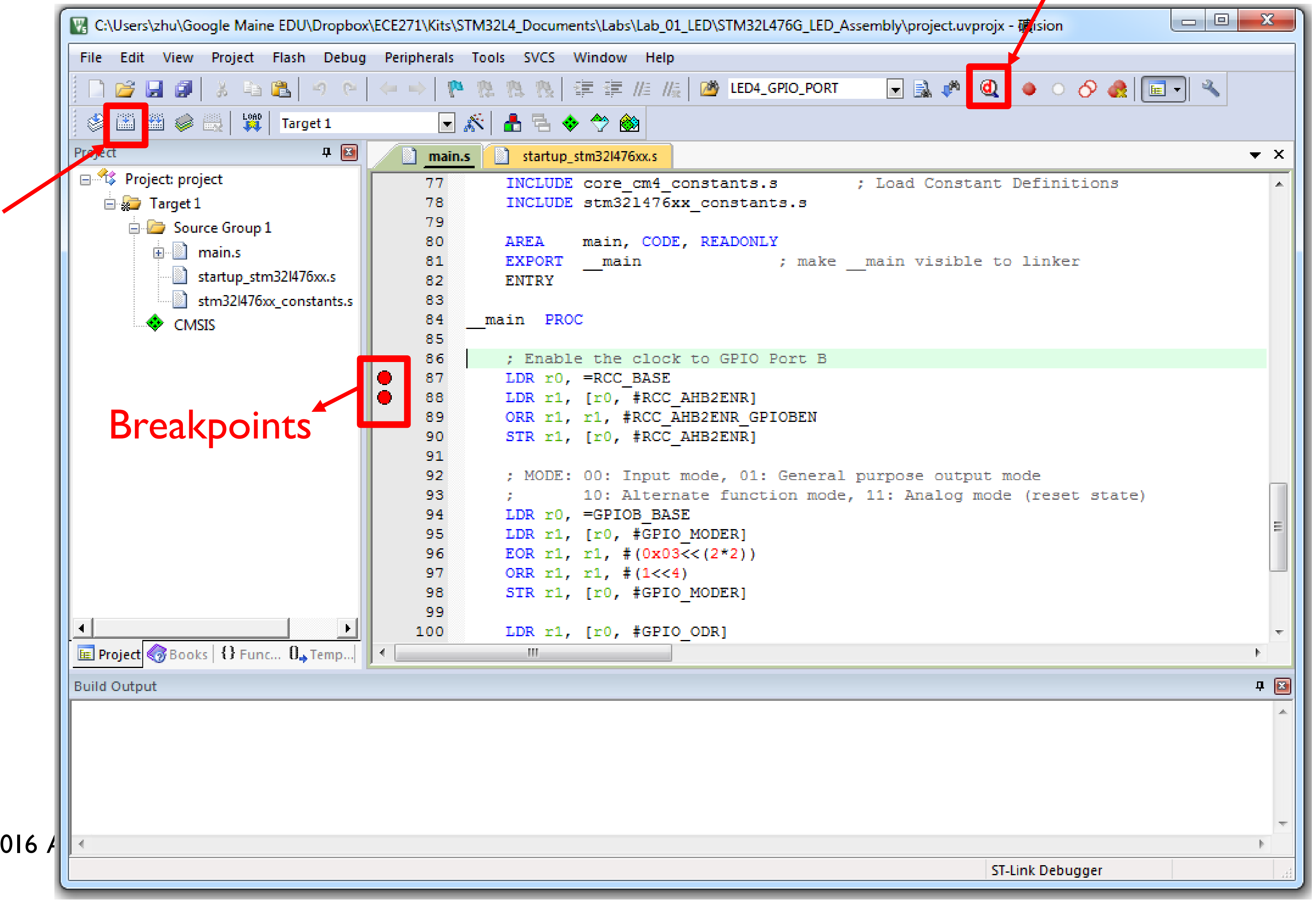
- Low cost
  - ~$20 each
- Hands-on experiences
  - develop and test real systems
- Rewarding and engaging
  - immediately enjoy the fruit of labor
- Convenient
  - mobile lab without time and location constrains
- Versatile
  - pins are extended for easy access



STM32L4 Discovery Kit @STMicroelectronics

Selecting a Platform: Software Component

- Keil uVision Development Tools

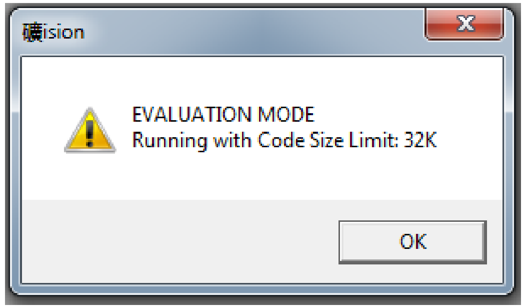


```
    INCLUDE core_cm4_constants.s        ; Load Constant Definitions
    INCLUDE stm32l476xx_constants.s

    AREA    main, CODE, READONLY
    EXPORT  __main               ; make __main visible to linker
    ENTRY

__main  PROC

    ; Enable the clock to GPIO Port B
    LDR r0, =RCC_BASE
    LDR r1, [r0, #RCC_AHB2ENR]
    ORR r1, r1, #RCC_AHB2ENR_GPIOBEN
    STR r1, [r0, #RCC_AHB2ENR]

    ; MODE: 00: Input mode, 01: General purpose output mode
    ;       10: Alternate function mode, 11: Analog mode (reset state)
    LDR r0, =GPIOB_BASE
    LDR r1, [r0, #GPIO_MODER]
    EOR r1, r1, #(0x03<<(2*2))
    ORR r1, r1, #(1<<4)
    STR r1, [r0, #GPIO_MODER]

    LDR r1, [r0, #GPIO_ODR]
```

Debug

Build

Breakpoints

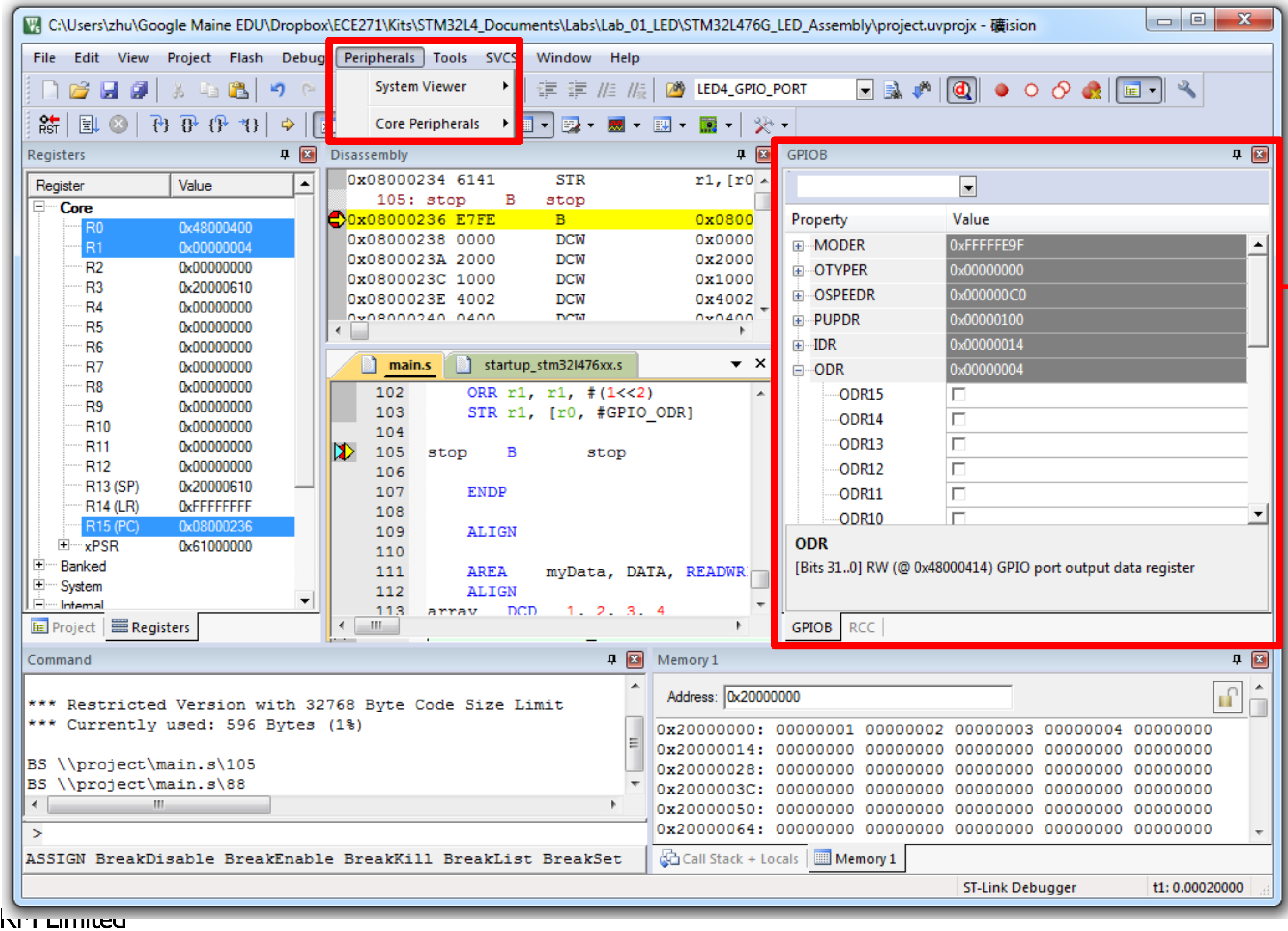


EVALUATION MODE
Running with Code Size Limit: 32K

But this has not been a problem.

Selecting a Platform: Software Component

- Keil uVision Development Tools



Register	Value
Core	
R0	0x48000400
R1	0x00000004
R2	0x00000000
R3	0x20000610
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000610
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x08000236
xPSR	0x61000000
Banked	
System	
Internal	

```
0x08000234 6141      STR        r1,[r0
   105: stop      B   stop
0x08000236 E7FE      B          0x0800
0x08000238 0000      DCW        0x0000
0x0800023A 2000      DCW        0x2000
0x0800023C 1000      DCW        0x1000
0x0800023E 4002      DCW        0x4002
0x08000240 0400      DCW        0x0400
```

```
102      ORR r1, r1, #(1<<2)
103      STR r1, [r0, #GPIO_ODR]
104
105  stop    B       stop
106
107      ENDP
108
109      ALIGN
110
111      AREA    myData, DATA, READWR
112      ALIGN
113  array   DCD   1, 2, 3, 4
```

Property	Value
MODER	0xFFFFFE9F
OTYPER	0x00000000
OSPEEDR	0x000000C0
PUPDR	0x00000100
IDR	0x00000014
ODR	0x00000004
ODR15	□
ODR14	□
ODR13	□
ODR12	□
ODR11	□
ODR10	□

ODR
[Bits 31..0] RW (@ 0x48000414) GPIO port output data register

```
*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 596 Bytes (1%)

BS \\project\main.s\105
BS \\project\main.s\88

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

Address: 0x20000000

```
0x20000000: 00000001 00000002 00000003 00000004 00000000
0x20000014: 00000000 00000000 00000000 00000000 00000000
0x20000028: 00000000 00000000 00000000 00000000 00000000
0x2000003C: 00000000 00000000 00000000 00000000 00000000
0x20000050: 00000000 00000000 00000000 00000000 00000000
0x20000064: 00000000 00000000 00000000 00000000 00000000
```

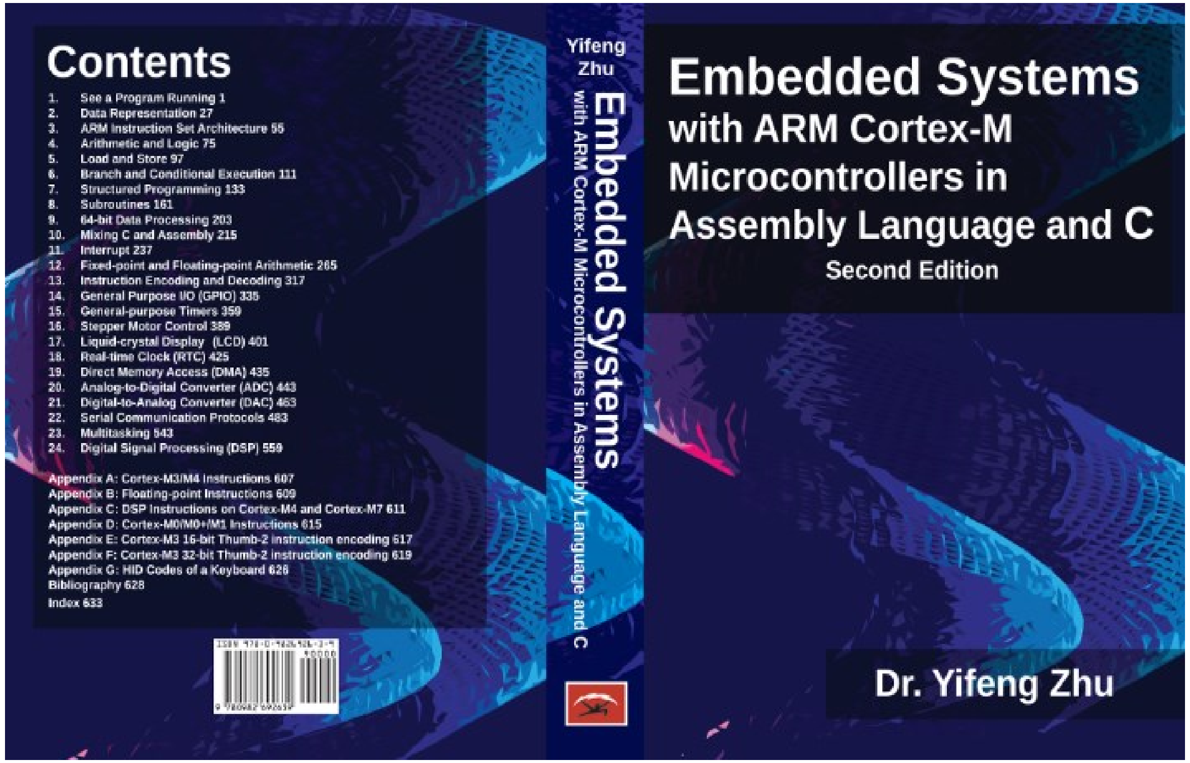
Monitor or modify peripheral registers

Students found this very helpful!

Free version limited the code size to 32 KB. But this has not been a problem.

Book

- Lab-centered learning
- State-of-the-art content
- Bare-metal programming in C and assembly to facilitate deep learning
- Line-by-line translation between C and Cortex assembly for most examples
- Mixture of C and assembly languages
- Suitable for all levels of undergraduate courses



Contents

Yifeng Zhu

Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C

Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C

Second Edition

Dr. Yifeng Zhu

Contact me if you want an evaluation copy!

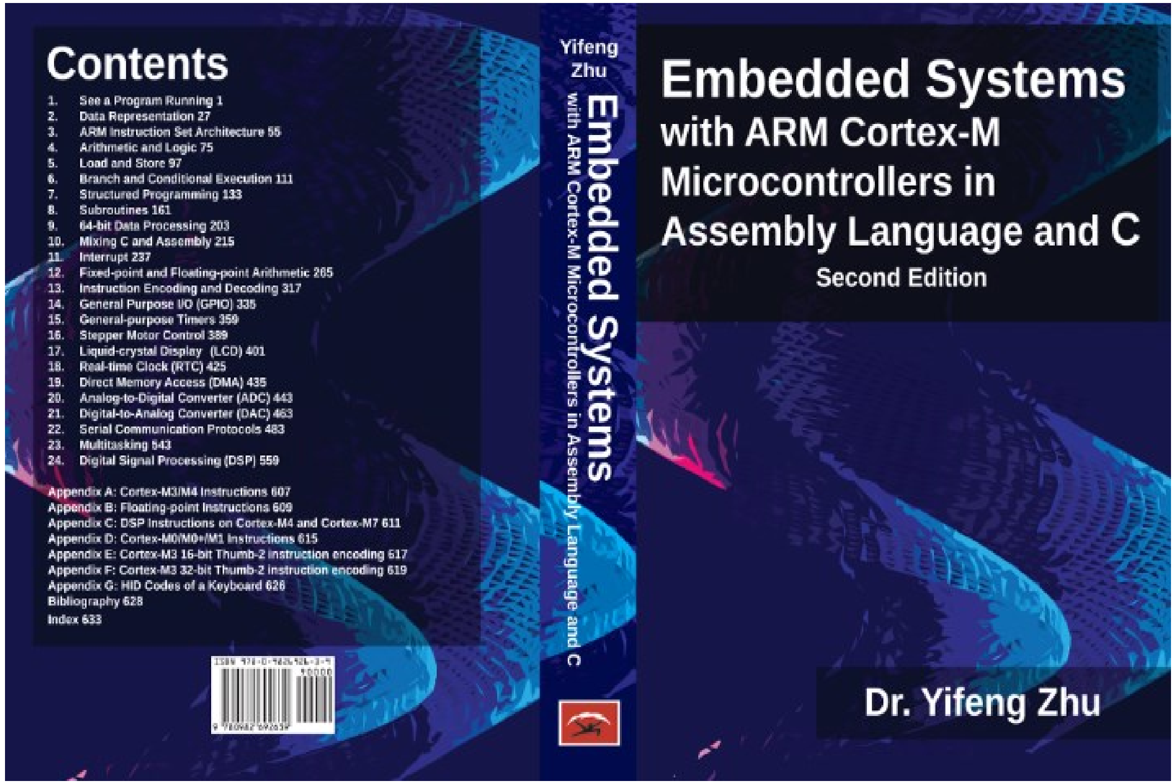
ISBN: 0982692633
660 pages, $69.50

USB drive contains:

- Lecture slides
- Lab description and solutions
- Instructor manual

Teaching Modules

- Data representation
- Assembly instructions and programming
- Mixing C and assembly
- Fixed-point & floating-point numbers
- FPU
- GPIO
- Interrupt service routines
- Timers
- DMA
- ADC and DAC
- I2C, SPI, UART, USB
- DSP



Contents

Yifeng Zhu

Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C

Embedded Systems
with ARM Cortex-M
Microcontrollers in
Assembly Language and C
Second Edition

Dr. Yifeng Zhu

Strike the balance between theoretical foundations and technical practices

Example Labs

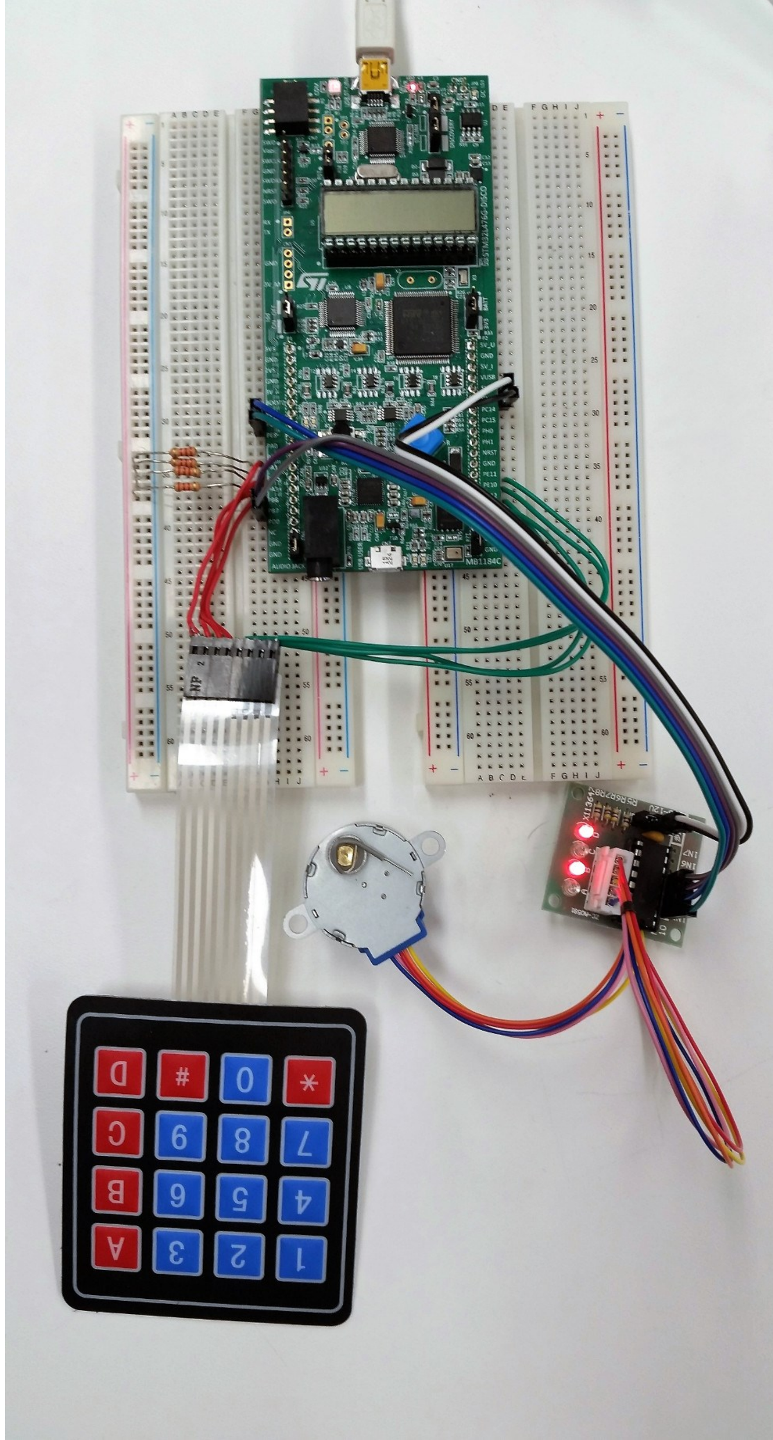
Lab descriptions and solutions are in the USB drive

Lower-division Course

1. Push buttons and LEDs
2. Matrix keypad
3. LCD display
4. Stepper motor
5. System timer (SysTick)
6. Timer PWM output
7. Timer input capture
8. ADC (polling)
9. DAC (polling)
10. Music synthesizing

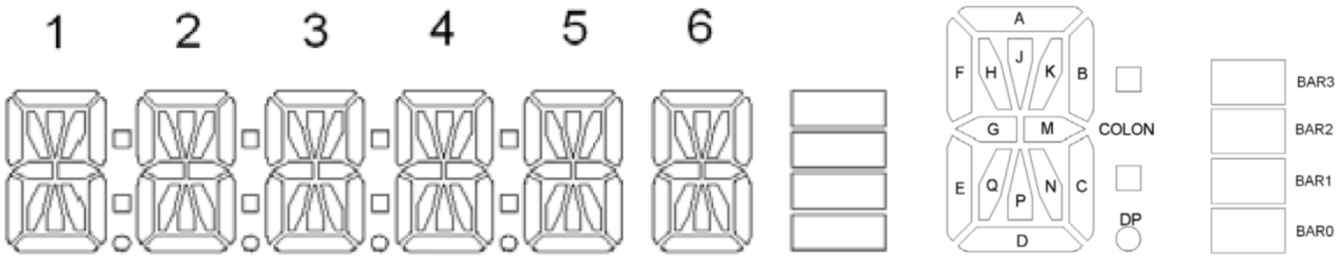
Upper-division Course

1. UART debugging
2. External interrupts
3. System timer (SysTick)
4. RGB LED Strip
5. ADC (DMA and/or interrupts)
6. DAC (DMA and/or interrupts)
7. DH22 temperature/humidity sensor
8. Gyro and accelerometer
9. nRF2401 wireless communication
10. Microphone & CODEC



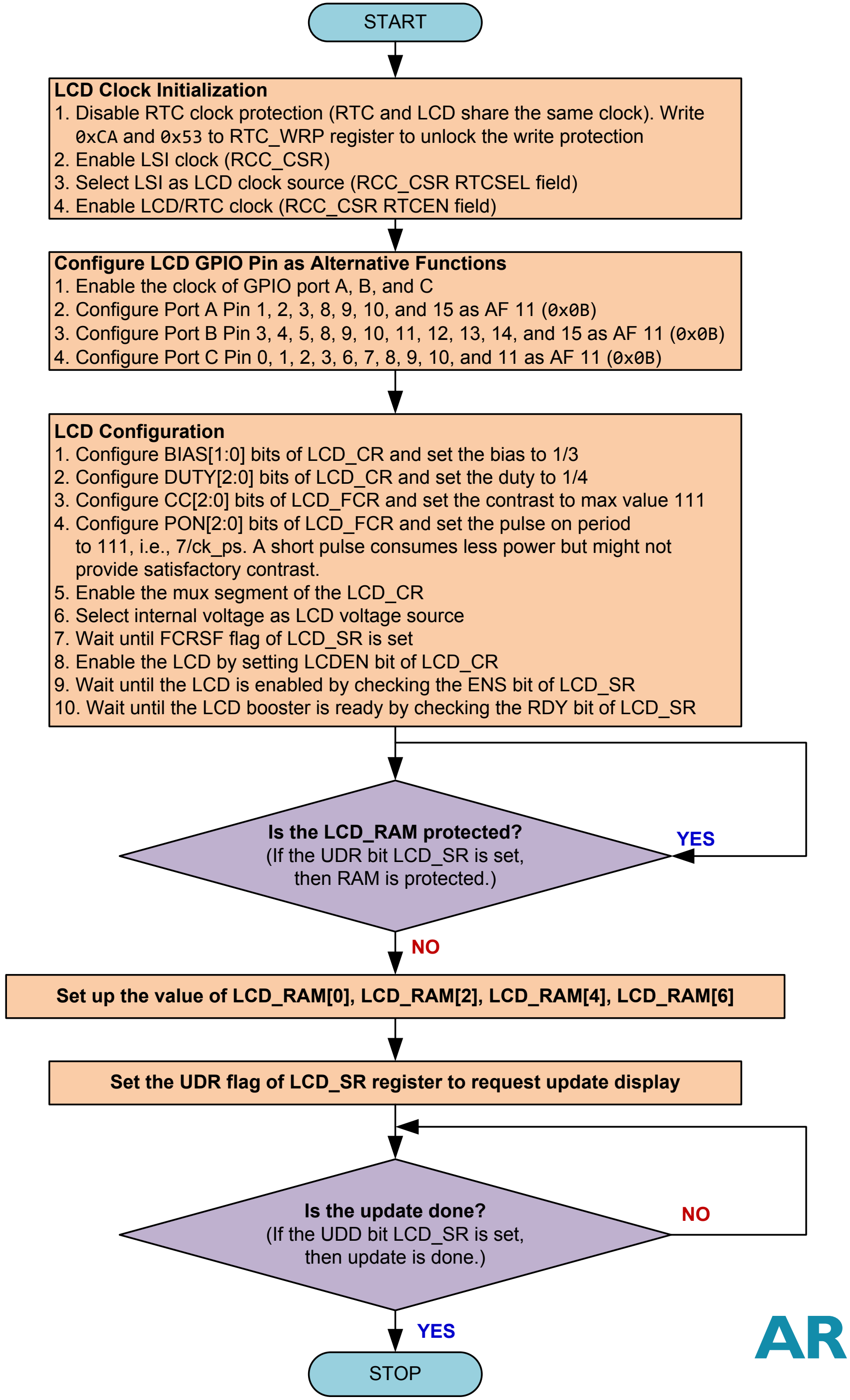
Example Lab: LCD Display

Write down your last name, and complete the following table.



Your Last Name: ______________________ (First Six Characters)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCD_RAM[0]	4E	4G	3M	3B		6G	5M	5B	1M	1B					6E		3E	3G	2M	2B			6B	6M		2E	2G	1E	1G			
LCD_RAM[1]	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨									5E	5G	4M	4B
	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
LCD_RAM[2]	4D	4F	3C	3A		6F	5C	5A	1C	1A					6D		3D	3F	2C	2A			6A	6C		2D	2F	1D	1F			
LCD_RAM[3]	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨									5D	5F	4C	4A
	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
LCD_RAM[4]	4P	4Q	3 Col	3K		6Q	3 Bar	5K	1 Col	1K					6P		3P	3Q	2 Col	2K			6K	1 Bar		2P	2Q	1P	1Q			
LCD_RAM[5]	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨									5P	5Q	4 Col	4K
	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
LCD_RAM[6]	4N	4H	3 DP	3J		6H	2 Bar	5J	1 DP	1J					6N		3N	3H	2 DP	2J			6J	0 Bar		2N	2H	1N	1H			
LCD_RAM[7]	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨									5N	5H	4 DP	4J
	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												



START

LCD Clock Initialization
1. Disable RTC clock protection (RTC and LCD share the same clock). Write `0xCA` and `0x53` to RTC_WRP register to unlock the write protection
2. Enable LSI clock (RCC_CSR)
3. Select LSI as LCD clock source (RCC_CSR RTCSEL field)
4. Enable LCD/RTC clock (RCC_CSR RTCEN field)

Configure LCD GPIO Pin as Alternative Functions
1. Enable the clock of GPIO port A, B, and C
2. Configure Port A Pin 1, 2, 3, 8, 9, 10, and 15 as AF 11 (`0x0B`)
3. Configure Port B Pin 3, 4, 5, 8, 9, 10, 11, 12, 13, 14, and 15 as AF 11 (`0x0B`)
4. Configure Port C Pin 0, 1, 2, 3, 6, 7, 8, 9, 10, and 11 as AF 11 (`0x0B`)

LCD Configuration
1. Configure BIAS[1:0] bits of LCD_CR and set the bias to 1/3
2. Configure DUTY[2:0] bits of LCD_CR and set the duty to 1/4
3. Configure CC[2:0] bits of LCD_FCR and set the contrast to max value 111
4. Configure PON[2:0] bits of LCD_FCR and set the pulse on period to 111, i.e., 7/ck_ps. A short pulse consumes less power but might not provide satisfactory contrast.
5. Enable the mux segment of the LCD_CR
6. Select internal voltage as LCD voltage source
7. Wait until FCRSF flag of LCD_SR is set
8. Enable the LCD by setting LCDEN bit of LCD_CR
9. Wait until the LCD is enabled by checking the ENS bit of LCD_SR
10. Wait until the LCD booster is ready by checking the RDY bit of LCD_SR

Is the LCD_RAM protected?
(If the UDR bit LCD_SR is set, then RAM is protected.)

YES

NO

Set up the value of LCD_RAM[0], LCD_RAM[2], LCD_RAM[4], LCD_RAM[6]

Set the UDR flag of LCD_SR register to request update display

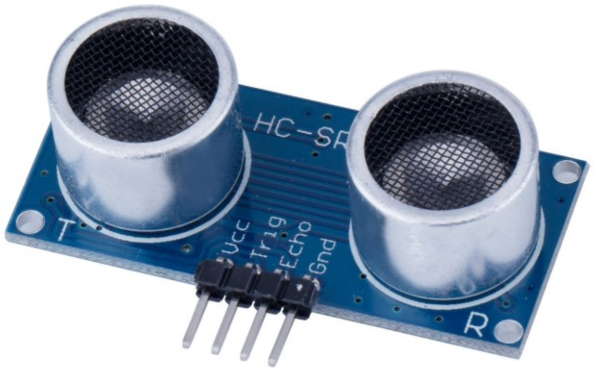
Is the update done?
(If the UDD bit LCD_SR is set, then update is done.)

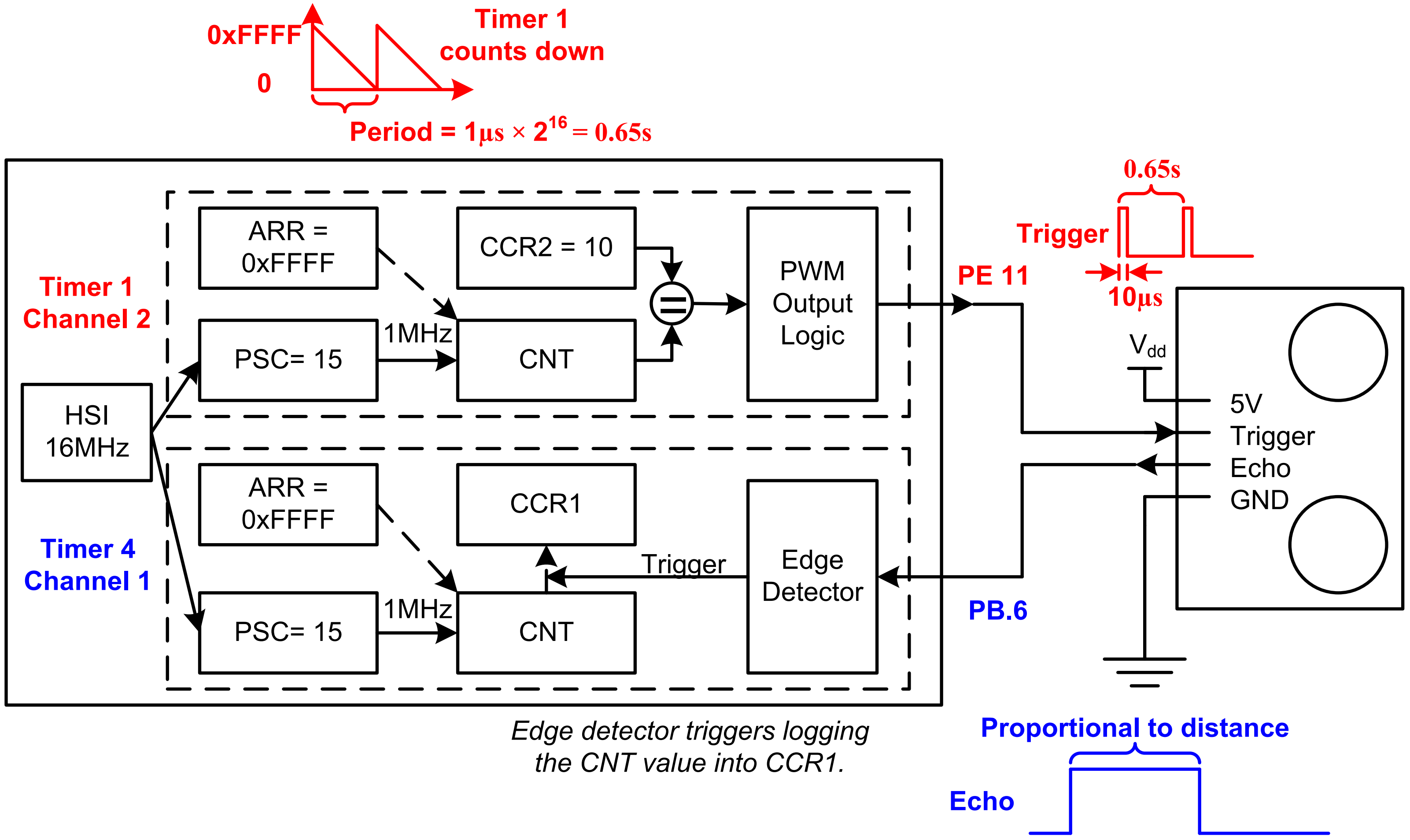
NO

YES

STOP

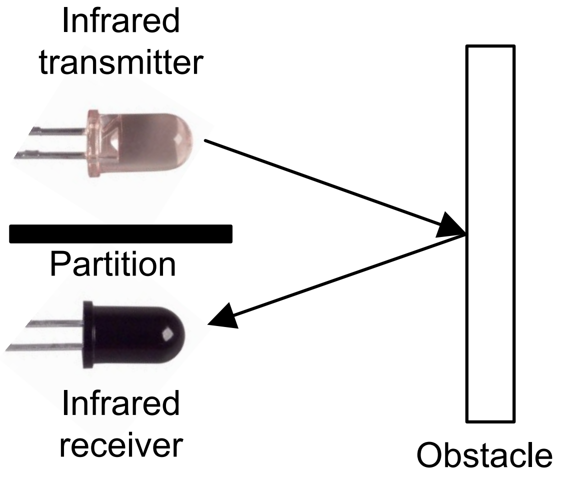
Example Lab: Ultrasonic Distance Measurement

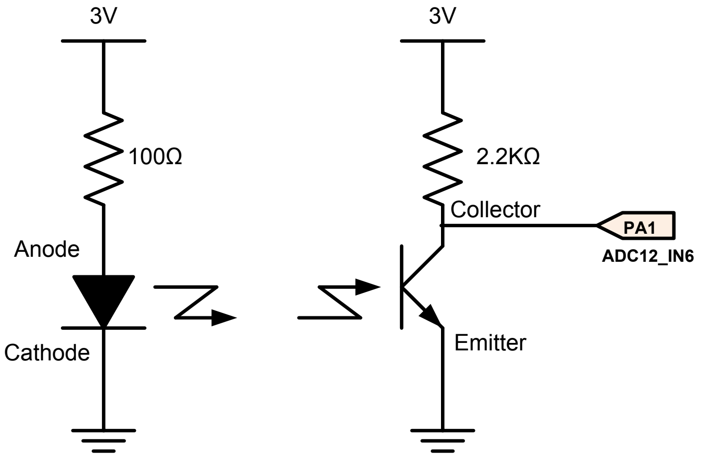


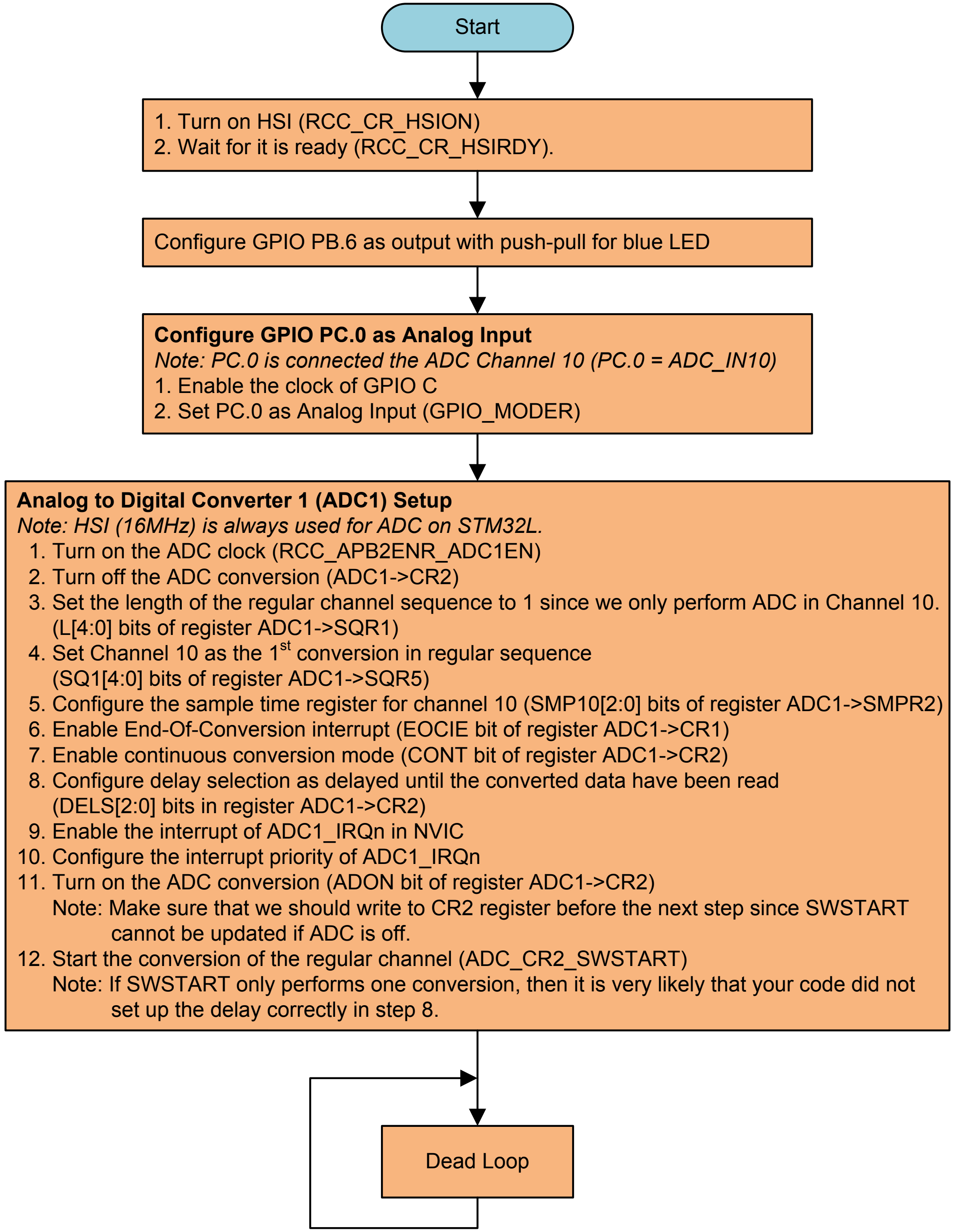


Edge detector triggers logging the CNT value into CCR1.

Example Lab: ADC







1. Turn on HSI (RCC_CR_HSION)
2. Wait for it is ready (RCC_CR_HSIRDY).

Configure GPIO PB.6 as output with push-pull for blue LED

Configure GPIO PC.0 as Analog Input

Note: PC.0 is connected the ADC Channel 10 (PC.0 = ADC_IN10)

1. Enable the clock of GPIO C
2. Set PC.0 as Analog Input (GPIO_MODER)

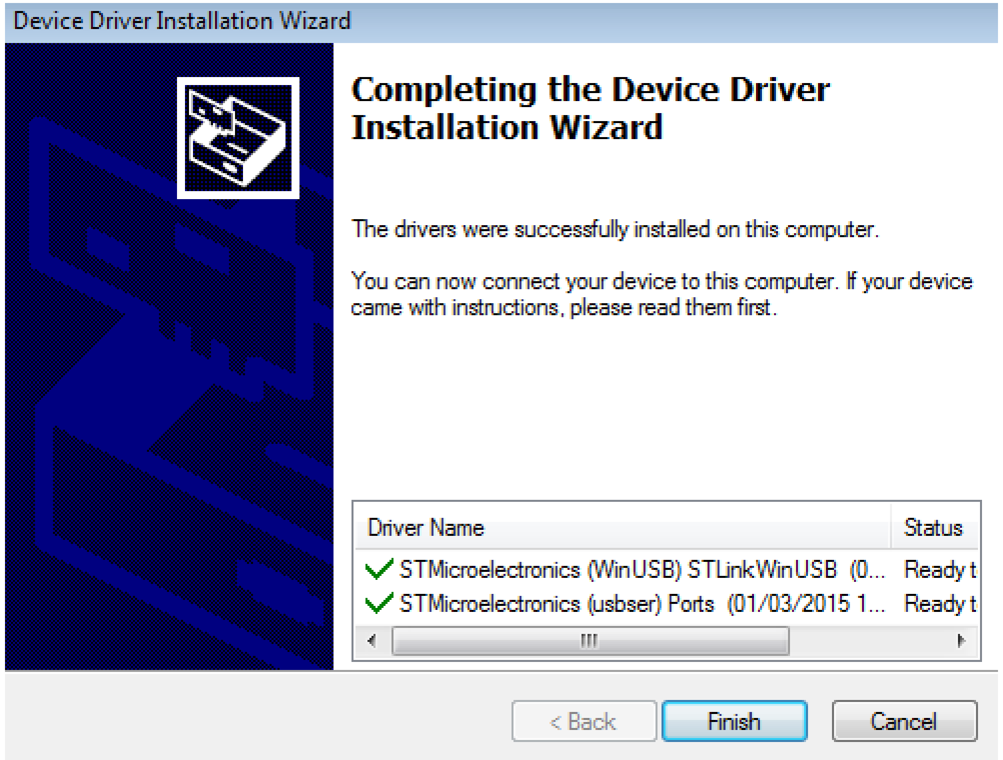
Analog to Digital Converter 1 (ADC1) Setup

Note: HSI (16MHz) is always used for ADC on STM32L.

1. Turn on the ADC clock (RCC_APB2ENR_ADC1EN)
2. Turn off the ADC conversion (ADC1->CR2)
3. Set the length of the regular channel sequence to 1 since we only perform ADC in Channel 10. (L[4:0] bits of register ADC1->SQR1)
4. Set Channel 10 as the 1st conversion in regular sequence (SQ1[4:0] bits of register ADC1->SQR5)
5. Configure the sample time register for channel 10 (SMP10[2:0] bits of register ADC1->SMPR2)
6. Enable End-Of-Conversion interrupt (EOCIE bit of register ADC1->CR1)
7. Enable continuous conversion mode (CONT bit of register ADC1->CR2)
8. Configure delay selection as delayed until the converted data have been read (DELS[2:0] bits in register ADC1->CR2)
9. Enable the interrupt of ADC1_IRQn in NVIC
10. Configure the interrupt priority of ADC1_IRQn
11. Turn on the ADC conversion (ADON bit of register ADC1->CR2)
    Note: Make sure that we should write to CR2 register before the next step since SWSTART cannot be updated if ADC is off.
12. Start the conversion of the regular channel (ADC_CR2_SWSTART)
    Note: If SWSTART only performs one conversion, then it is very likely that your code did not set up the delay correctly in step 8.

Install USB Driver

- Install ST-Link USB device driver
  - Go to the directory C:\Keil_v5\ARM\STLink\USBDriver and run stlink_winusb_install.bat



Device Driver Installation Wizard

Completing the Device Driver Installation Wizard

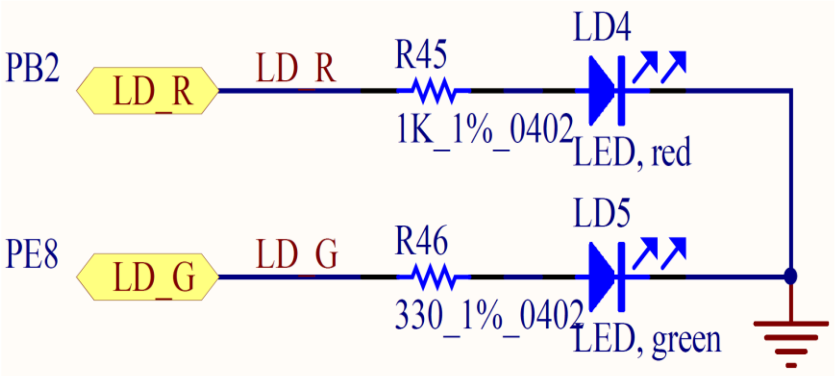
The drivers were successfully installed on this computer.

You can now connect your device to this computer. If your device came with instructions, please read them first.

Driver Name	Status
✓ STMicroelectronics (WinUSB) STLinkWinUSB (0...	Ready t
✓ STMicroelectronics (usbser) Ports (01/03/2015 1...	Ready t

Hands-on Lab #1

Light up an LED in 100% assembly



Pre-Lab Assignment

1. **Enable the clock of GPIO Port A (for joy stick), Port B (for Red LED) and Port E (for Green LED)**

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB2ENR														RNGEN		AESEN			ADCEN	OTGFSEN					GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
Mask																																
Value																																

a. **Configure PB 2 as Output**

GPIO Mode: Input (00), Output (01), Alternative Function (10), Analog (11, default)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
Mask																																
Value																																

b. **Configure PB 2 Output Type as Push-Pull**

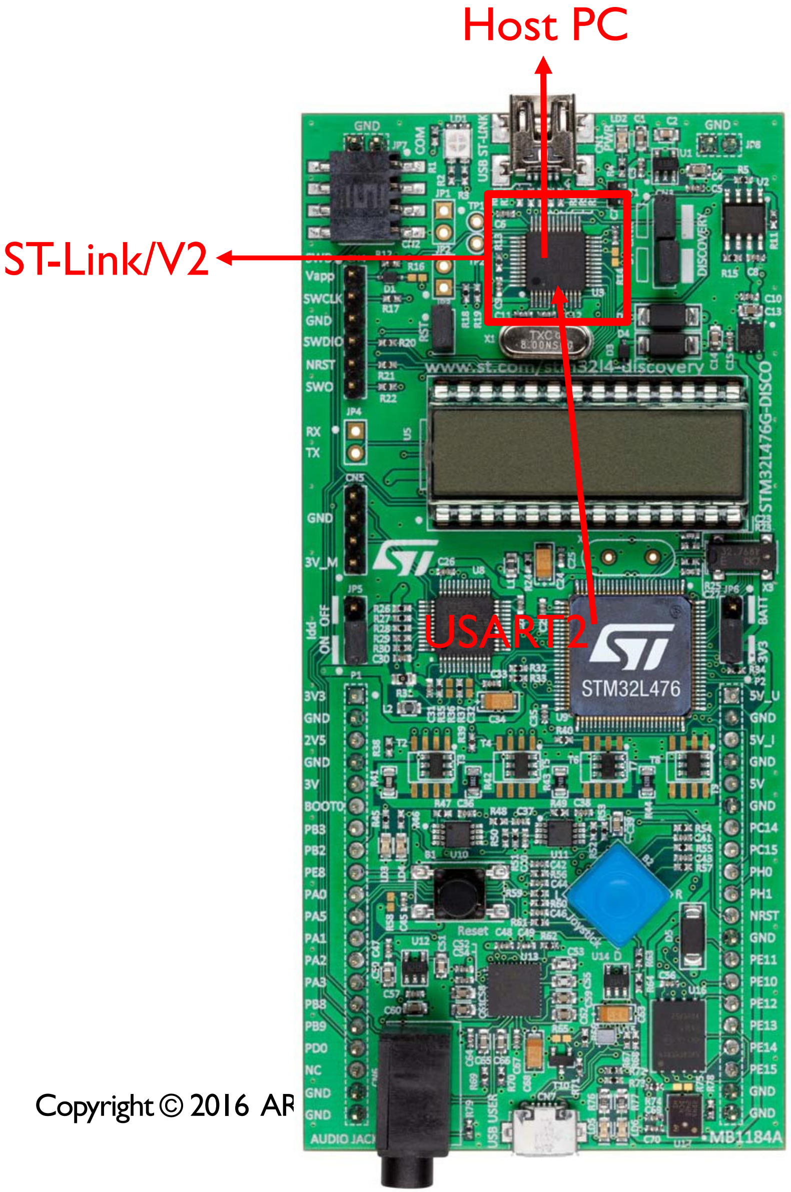
Push-Pull (0, reset), Open-Drain (1)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OTYPER	Reserved																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
Mask																																
Value																																

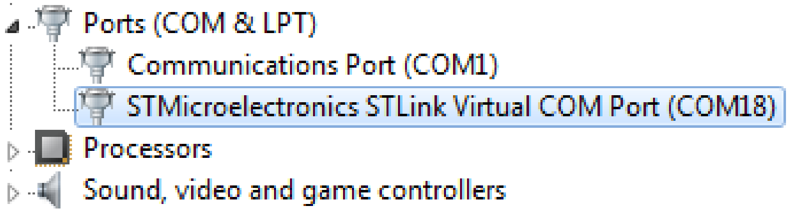
Hands-on Lab #2

Printing messages via UART through ST-Link V2

No extra cable or devices are required!



- USART2 (PD5 TX, PD6 RX) is connected to ST-Link's UART
- ST-Link has virtual-COM port
- Setting USART2 as Asynchronous, 9600, 8/N/1
- Need terminal emulator (such as Tera Term for Windows, and MacWise for Mac)



```
◢ Ports (COM & LPT)
    Communications Port (COM1)
    STMicroelectronics STLink Virtual COM Port (COM18)
▷ Processors
▷ Sound, video and game controllers
```

Find the com port # from device manager

More Resources

- Book Website: FAQ, lab parts, example codes
  - http://web.eece.maine.edu/~zhu/book/
- Tutorial 1: Create a project in Keil
  - https://www.youtube.com/watch?v=0t_Myn4UYUw
- Tutorial 2: Debug in Keil
  - https://www.youtube.com/watch?v=w4gPcYRk9o8
- Tutorial 3: Clock configuration of STM32L4 processors
  - https://www.youtube.com/watch?v=w4gPcYRk9o8
- Tutorial 4: Printing messages via UART through ST-Link V2.1
  - https://www.youtube.com/watch?v=u9vUyRjtG3Y

Conclusions

- Teaching students both C and Assembly is important
- Emphasize the balance between theoretical foundations and technical practices
- Hand-on experiences based on lab-in-a-box platforms are effective

Acknowledge

Thank ARM and STMicroelectronics
for sponsoring this workshop!