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

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](http://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);

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

Learning to see the forest and the trees!

- Pros
  - Simplify implementation
  - Better portability
  - Many examples

- Cons
  - Very complex to understand
  - Cannot meet students’ curiosity
  - Does not facilitate deep learning
#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), 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-&gt;MODER &amp;= ~(3&lt;&lt;(2*LED_PIN));
GPIOB-&gt;MODER |&lt;= 1&lt;&lt;(2*LED_PIN); // Output(01)

LED_PIN EQU 2

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

Diming LED by using timer PWM output

Enable Peripheral Clocks via RCC Registers
1. Enable the clock of GPIO port B
2. Enable the clock of timer 4

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_AFRH)
3. Set the pin as push-pull mode with no pull-up pull-down (SPI0_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_CCR1)
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)
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

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 Discovery Kit @STMicroelectronics
STM32L476G 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
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

- 9-axis motion sensor (underneath LCD)
- Joystick
- Reset
- LEDs
- Audio Codec
- Audio Connector
- Flash
- Microphone
- USB OTG
Selecting a Platform: Software Component

- Keil uVision Development Tools

But this has not been a problem.
Selecting a Platform: Software Component

- Keil uVision Development Tools

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.
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

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

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.

<table>
<thead>
<tr>
<th>Your Last Name: ____________________ (First Six Characters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD_RAM[0]</td>
</tr>
<tr>
<td>LCD_RAM[2]</td>
</tr>
<tr>
<td>LCD_RAM[6]</td>
</tr>
</tbody>
</table>

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 (0x88)
3. Configure Port B Pin 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, and 15 as AF 11 (0x88)
4. Configure Port C Pin 0, 1, 2, 3, 6, 7, 8, 9, 10, and 11 as AF 11 (0x88)

LCD Clock Initialization
1. Disable RTC clock protection (RTC and LCD share the same clock). Write 0x04 and 0x93 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)

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 tick 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.)

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

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

Set the UDR flag of LCD_SR register to request update display

START
NO
NO
YES
STOP
Example Lab: Ultrasonic Distance Measurement

Timer 1
Channel 2

HSI 16MHz

Timer 4
Channel 1

ARR = 0xFFFF
PSC = 15
1MHz

ARR = 0xFFFF
PSC = 15
1MHz

CCR2 = 10
CNT

CCR1
CNT

PWM Output Logic

Edge Detector

Trigger

Period = 1μs × 2^16 = 0.65s

Timer 1 counts down

Edge detector triggers logging the CNT value into CCR1.
Example Lab: ADC

Analog to Digital Converter 1 (ADC1) Setup

1. Start Configure GPIO PC
2. 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)

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

Start

1. Turn on HSI (RCC_CR_HSION)
2. Wait for it is ready (RCC_CR_HSIRDY).
3. Configure the sample time register for channel 10 (SMP10[2:0] bits of register ADC1->SMPR2)
4. 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)
5. Enable the interrupt of ADC1_IRQn in NVIC
6. Configure delay selection as delayed until the converted data have been read (DELS[2:0] bits in register ADC1->CR2)
7. Enable continuous conversion mode (CONT bit of register ADC1->CR2)
8. Configure the interrupt priority of ADC1_IRQn
9. Set Channel 10 as the 1st conversion in regular sequence (SQ1[4:0] bits of register ADC1->SQR5)
10. Enable End-Of-Conversion interrupt (EOCIE bit of register ADC1->CR1)
11. Configure GPIO PB
    1. Turn on HSI (RCC_APB2ENR_ADC1EN)
    2. Turn off the ADC conversion (ADC1->CR2)
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`
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)

<table>
<thead>
<tr>
<th>Register</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHB2ENR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mask</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Configure PB 2 as Output

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

<table>
<thead>
<tr>
<th>Register</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mask</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b. Configure PB 2 Output Type as Push-Pull

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

<table>
<thead>
<tr>
<th>Register</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTYPEP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mask</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Printing messages via UART through ST-Link V2

- 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)

No extra cable or devices are required!

Find the com port # from device manager
More Resources

- Book Website: FAQ, lab parts, example codes

- 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!