Introduction to embedded systems
What are embedded systems?
- Definition
- Application areas
- Classification
- Common architecture

What are the basic elements of the embedded systems?
- Common embedded systems hardware architectures
- Embedded software development flow

How to write program for embedded device?
- “Hello World” from embedded systems 😊

Life as an embedded system developer?
Embedded systems
- Definition, architectures and classification -
Embedded systems are everywhere around us...

One of the fastest growing areas in the world!

Embedded systems makes the world smarter and more advanced!
An **embedded system** can be broadly defined as a device that contains tightly coupled hardware and software components which perform a single function, forms part of a larger system, is not intended to be independently programmable by the user, and is expected to work with minimal or without human interaction.

An embedded system is a combination of computer hardware and software, either fixed in capability or programmable, designed for a specific function or functions within a larger system.

Embedded system is a special-purpose computing device designed to perform dedicated functions. An embedded system consists of its hardware and software.
Embedded system is a special-purpose computing device designed to perform dedicated function.

**Input Devices**
- Keyboard Button
- Steering wheel position
- GSM Engine
- Microphone

**Output Devices**
- LCD Display
- Servo Engine
- Speaker

**Diagram:**
- Embedded board
  - MCU
  - Input devices
  - Output devices
- System software
  - Middleware
  - Device drivers
- Application software

**Environments:**
- Analog Environment
- Digital Environment
Embedded system classifications

- **Stand-Alone Embedded systems**

- **Real-Time Embedded systems**
  - $t = T$
  - $t = T + \Delta t$
  - *Soft Real-Time*
  - *Hard Real-Time*

- **Network Embedded systems**
Life as an embedded system developer

• Embedded system developer is engineer who must understand the whole computer system!

• Programming at the lowest level – the closest to the hardware
  ➢ Understanding digital and analog electronics
  ➢ Understanding big system schematic pictures

• Writing efficient and reliable code
  ➢ Embedded platforms have limited resources

• Full system control, access to each bit in the system
  ➢ A lot of responsibilities
  ➢ Big space for errors

• Ability to easily adapt to new development environments
“Hello World“ program

For STM32L476RG MCU on NUCLEO development board

Create software for STM32L476RG MCU available on STM32 Nucleo development board which uses one button and one LE Diode. Diode state changes each time (from OFF to ON) when user press blue button on the NUCLEO board.

... but first!
Embedded systems
- Hardware -
Low level programming – programming closest to the hardware

- Before writing software for embedded system device, you must become familiar with the architecture of the platform for which you want to write the software
  - Read datasheets, user manuals, reference manuals
    - Usually available on the manufacturer’s website in “support” section
  - Try to understand MCU architecture
  - Examine MCU processor properties and features
  - Deeply explore system memory map
  - Study the external peripherals
Embedded systems - Hardware

EMBEDDED BOARD COMMON ARCHITECTURE

EMBEDDED SYSTEM

GLOBAL I/O PINS

INPUT
Buttons, Switches

INPUT
Sensor

OUTPUT
Led diode

OUTPUT
LCD Display

DEBUG SUPPORT
UART

Development / Embedded Board
Embedded systems - Hardware

NUCLEO-L476RG Development board

- USER BUTTONS
- USER LEDS
- GLOBAL PURPOSE IO PINS
- MCU
- DEBUG SUPPORT

HARDWARE

EMBEDDED BOARD

- INPUT DEVICES
- OUTPUT DEVICES
- MCU
Embedded systems - Hardware

MCU COMMON ARCHITECTURE

- MCU is a digital system

- CPU
  Fetch-decode-execute

- Memory
  Store Data and Code

- Peripherals
  Communicate with the outside world or facilitate dedicated actions
Embedded systems - Hardware

STM32L476RG MCU Memory Map
Embedded systems - Hardware

STM32L476RG MCU Block Diagram

CPU

BUSES

PERIPHERALS

MEMORY
Embedded systems - Hardware

NUCLEO-L476RG Development board hardware schematic

- Examine schematic in order to see how MCU communicates with the board environment

```
<table>
<thead>
<tr>
<th>BUTTON STATE</th>
<th>PC_13 PIN VALUE</th>
<th>DIODE STATE</th>
<th>PA_5 PIN VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressed</td>
<td>PC_x = 31 14 13 12 0</td>
<td>GPIOC_IDR</td>
<td>PA_x = 31 6 5 4 0</td>
</tr>
<tr>
<td>Released</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Interface to digital world

```
<table>
<thead>
<tr>
<th>GPIOA_ODR</th>
</tr>
</thead>
<tbody>
<tr>
<td>X ... X 1 X ... X</td>
</tr>
</tbody>
</table>
```
Embedded systems - Hardware

NUCLEO-L476RG I/O Data register position and access

HARDWARE
EMBEDDED BOARD
- INPUT DEVICES
- OUTPUT DEVICES
- MCU

CPU

AHB

GPIOC

GPIOC_IDR

BUTTON STATE

GPIOA

GPIOA_ODR

DIODE STATE

AHB2 BUS MM

PERIPHERY MM

0x5006 0BFF
0x5FFF FFFF
0x4800 0000

Reserved
AHB2

Reserved
AHB1

APB2

Reserved
APB1

0x4000 0000

0xFFFF FFFF

GPIOx_ODR

GPIOx_IDR

0x00

0x10

0x14

0x2C

GPIOx MEMORY MAP

GPIOA

GPIOB

GPIOC

GPIOD

SYSTEM MM

PERIPHERALS

0x0000 0000

0x4000 0000

0x5006 0BFF
0x4800 0000

Reserved
APB1

Reserved
APB2

Reserved
AHB1

Reserved
AHB2

GPIOx_IDR address is equal to?
GPIOx_ODR address is equal to?
Embedded systems
- Software -
Embedded systems - Software

Driving force of an embedded system

➢ Embedded systems programming is like any other software development, plus special consideration for resource constraints

1. CPU performance constraints
2. Power constraints
3. Memory constraints

➢ Application specific software
  • “Endless” running software
    - Reliability
    - Safety

➢ Usually written in C programming language
  • Different IDE
    - Keil, Eclipse, STMCubeIDE

➢ Embedded software or firmware – Where is a difference?
Embedded systems - Software

Software Project Development Process Flow

1. **Create Project**
2. **Setup Project**
3. **Create Startup File**
4. **Edit/Create Source/Assembler Files**
5. **Compile and Link**
6. **Build Success?**
   - **Yes**: **Download Image to MCU Flash Memory**
   - **No**: **Yes**
7. **Start Debugging**
8. **Program Works as Expected?**
   - **Yes**: **Software is Done**
   - **No**: **Software is Done**

- Project working directory on host side is set
- Project hardware platform, optimization levels, file types … are set
- Assembly startup file is ready
- Assembly and source files are ready
- Target Image file is ready

**NO**

**YES**
Embedded systems - Software

Embedded Software Integrated Development Environment

➢ Embedded system software development tool
   • Application on host side (PC) which allows us to create and setup software project properties for our target embedded device

➢ ESIDE Integrates:
   • Text editor
   • Toolchain
     • (Cross) Assembler
     • (Cross) Compiler
     • Linker
     • Debugger
     • Additional Libraries
   • Loader tool
   • Debugging tool
Embedded systems - Software

Common embedded device start procedure

- What steps does embedded device perform after power on or reset
- BOOT Sequence:

  1. CPU PC register is initialized to predefined value (for example 0x00000000)
  2. Instruction from address 0x00000000 is loaded and it is most commonly JUMP instruction which transfers execution to the start up code
  3. Start up code starts to execute

**SYSTEM MEMORY MAP**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x00000004</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x00000008</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x08004000</td>
<td>0x08004000</td>
</tr>
<tr>
<td>0x08004004</td>
<td>0x08004000</td>
</tr>
<tr>
<td>0x08004008</td>
<td>0x08004000</td>
</tr>
<tr>
<td>0x20000000</td>
<td>0x20000000</td>
</tr>
<tr>
<td>0x20000004</td>
<td>0x20000000</td>
</tr>
<tr>
<td>0x20000008</td>
<td>0x20000000</td>
</tr>
</tbody>
</table>

**SOFTWARE**

- APPLICATION SOFTWARE
- SYSTEM SOFTWARE
- MIDDLEWARE
- DEVICE DRIVERS

**IR**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x08000000</td>
<td>0x08000000</td>
</tr>
</tbody>
</table>

**SP**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>

**PC**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>

0xnnnnnnnnn - UNDEFINE VALUE
Embedded systems - Software

Common startup code flow sequence

➢ Startup code is small block of code that execute before our main program in order to properly initialize and prepare system for proper main program execution

➢ Start up code usually performs following steps
  • Disable all interrupts
  • Copy initialized data from flash/rom to RAM
  • Clear (zeros) unutilized data area
  • Allocate space and initialize STACK area
  • Copy STACK start address to the stack pointer register
  • Enable interrupts
  • Jump to main function

➢ Most cross-compilers for embedded systems already have this file. It is named as startup.asm, crt0.s, ...
**Embedded systems - Software**

Software source code development process stages

- Write code on HOST side for TARGET device
  - use some of the available ESIDEs

- Development stages

1. **Develop Software model**
   - Analyzes the requirements specification and presents an abstraction model describing the embedded software structure and behavior.
     - FSM
     - Program Flow diagram
     - Software layers block diagram
     - Choose the most suitable software flow

2. **Map model to the code**

3. **Write code on host side**

- Development result is

**TARGET IMAGE FILE**
**Embedded systems - Software**

„main“ source code development for our project example

1. **Develop Software model**

   ![Diagram](image)

   **INIT HARDWARE**
   - hardware initialization (clocks and peripherals initialization)

   **READ BUTTON STATUS**
   - read pin status from GPIOC_IDR register and return corresponding result

   **BUTTON PRESSED?**
   - if (ButtonStatus == PRESSED)
     - Change Diode Status
   - NO

   **CHANGE DIODE STATUS**
   - toggle pin value in GPIOA_ODR

2. **Map model to the code**

   ```c
   int main(void)
   {
     ButtonStatus ButtonStatus = RELEASED;
     InitHardware();
     while (1)
     {
       ButtonStatus = ReadButtonStatus();
       if(ButtonStatus == PRESSED){
         ChangeDiodeStatus();
       } NO
       YES
     }
   }
   
   typedef enum{
     PRESSED, RELEASED
   } ButtonStatus;
   
   ButtonStatus ReadButtonStatus(){
     if(HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13) == 0) return PRESSED;
     return RELEASED;
   }
   
   void ChangeDiodeStatus(){
     HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
   }
   ```

3. **Write code on host side**

   - INIT HARDWARE
   - READ BUTTON STATUS
   - CHANGE DIODE STATUS
Embedded systems - Software

Build process

➢ From user code (assembly and C code) create executable code which will be loaded to the target memory

➢ Compiler
  • Translate programs written in some human-readable language into the equivalent set of opcodes for particular processor
  • Cross-compiler
    ❑ Code is written on a different platform and than compiled for embedded platform
  • Compilation result is object file which corresponds to the dedicated source file

➢ Assembler
  • “Compiler for assembly language files”

➢ Linker
  • Combine object files and locate them in memory

➢ Result

TARGET IMAGE FILE
Embedded systems - Software

Build process and programming

- test1.c
- test2.c
- test3.asm

- test1.o
- test2.o
- test3.o

- platform.ld

- Image.map
- Image.elf

- Image.elf

- Flash Loader Tool

- Flash

- MCU

- HOST

- HOST

SOFTWARE

APPLICATION SOFTWARE

SYSTEM SOFTWARE

- Middleware
- Device Drivers
Conclusion

➢ Embedded systems are perspective and fast growing area in past few years

➢ Equal hardware and software skills are necessary to create reliable embedded system’s software

➢ There are a lot of projects that involve the development of embedded software
Questions?

If you don’t ask, you will not learn.
Thank you!

www.thevtool.com