Embedded and System Software Introduction

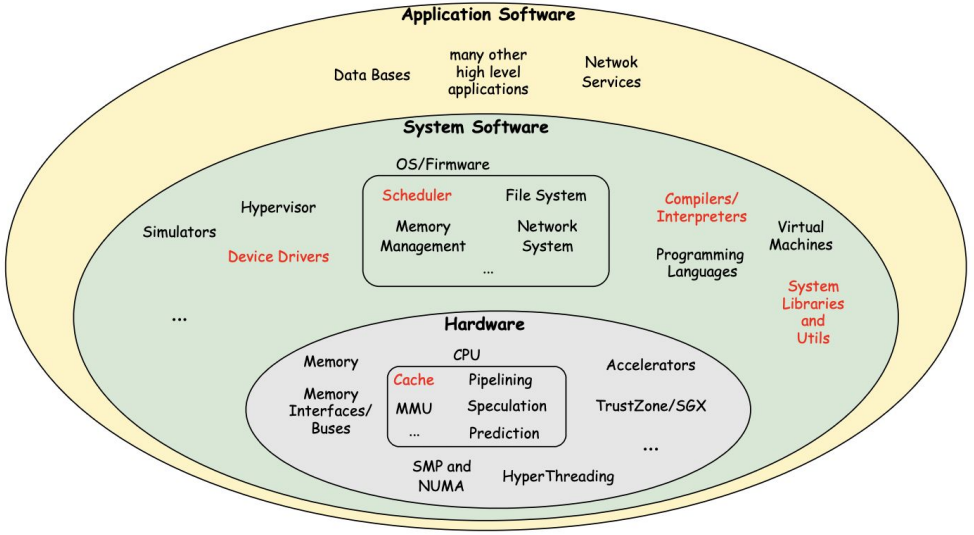
- О чем наш курс и кому он нужен
- Знакомство
- Об отладочной плате на которой будут проводиться лабораторные работы
- Небольшой пример
- Задание

Что такое системное программирование ?

System Software – ПО необходимое для запуска Application Software: Операционные системы и различные Firmware, компиляторы, драйвера операционных систем, симуляторы.

Application Software – ПО которое использует конечный пользователь для своих нужд: Word/Excel/etc, IDE, Video/Audio editors and players.

System Software Engineer knowledge	Application Software Engineer knowledge
Как работает железо	Прикладной язык программирования
Как работает операционная система, сетевой стэк	Алгоритмы/Структуры данных
Как работают компиляторы и интерпретаторы, во что компилируется ваш код	Какой то собственный Framework для разработки вашего Application Software

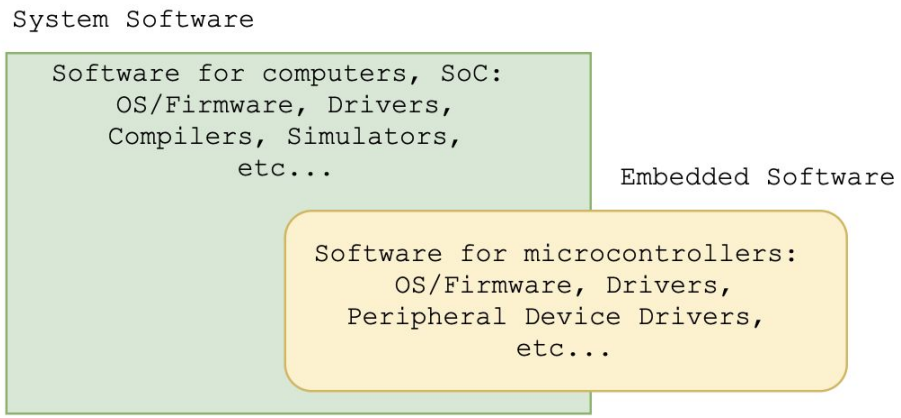


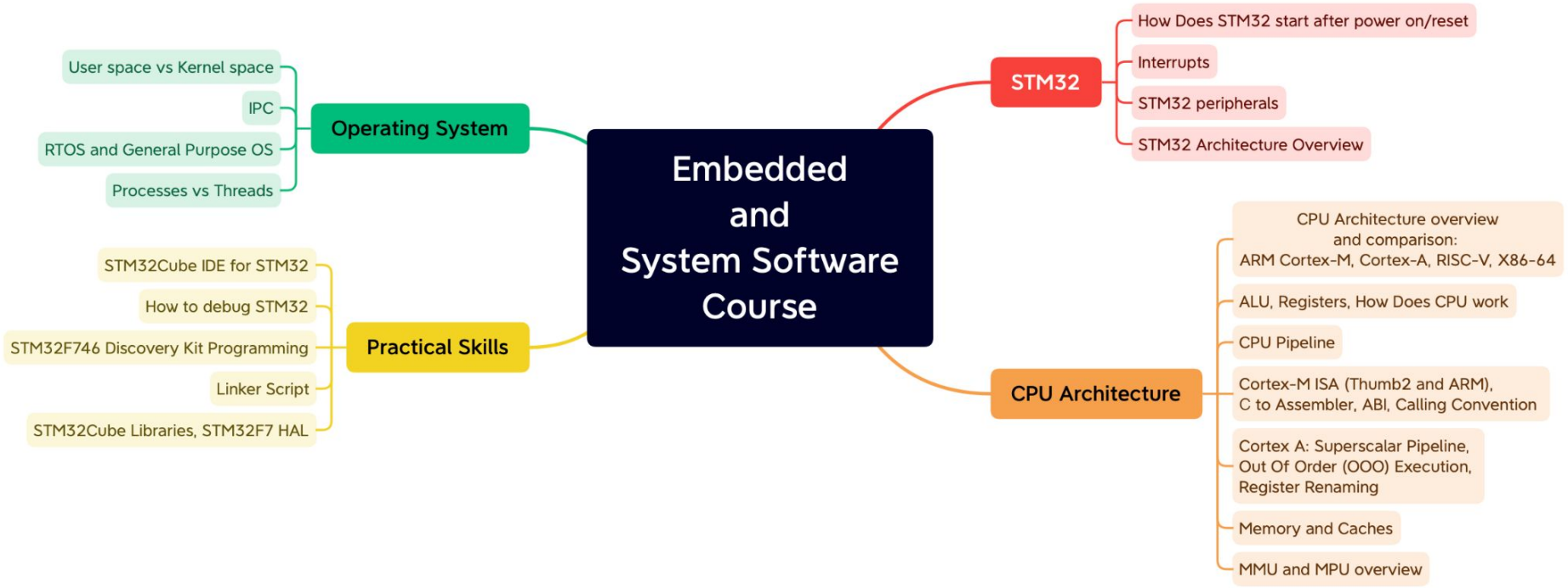
System Software vs Application Software

Что такое Embedded Software ?

Embedded Software – ПО для управления устройствами не являющимися компьютерами, телефонами. Например это ПО для управления: медицинским оборудованием, потребительской электроникой, автомобильным оборудованием и т.д.

System Software	Embedded Software
System Software для компьютеров, телефонов	Embedded Software для устройств управления
SoC (System on chip) и компьютеры	Микроконтроллеры
General Purpose OS	Embedded OS, RTOS
	Часто требуется дополнительное знание схемотехники




Embedded and System Software Course
Operating System
User space vs Kernel space
IPC
RTOS and General Purpose OS
Processes vs Threads
Practical Skills
STM32Cube IDE for STM32
How to debug STM32
STM32F746 Discovery Kit Programming
Linker Script
STM32Cube Libraries, STM32F7 HAL
STM32
How Does STM32 start after power on/reset
Interrupts
STM32 peripherals
STM32 Architecture Overview
CPU Architecture
CPU Architecture overview and comparison: ARM Cortex-M, Cortex-A, RISC-V, X86-64
ALU, Registers, How Does CPU work
CPU Pipeline
Cortex-M ISA (Thumb2 and ARM), C to Assembler, ABI, Calling Convention
Cortex A: Superscalar Pipeline, Out Of Order (OOO) Execution, Register Renaming
Memory and Caches
MMU and MPU overview

Компании	Что делать
Intel	Симуляторы (Simics), компиляторы, Linux kernel, Linux Drivers, Zephyr RTOS, BIOS, etc…
Samsung	Компиляторы, Tizen, TizenRT, Firmware, etc…
Huawei	Компиляторы, Операционные системы, Firmware для базовых станций, etc…
Yandex	ПО для беспилотных авто, Yandex devices
Sber	Робототехническая лаборатория, ПО для Sber Devices
Множество других компаний поменьше	ПО для управления различными устройствами: газовые корректора, телеметрия, etc…

Антонов Александр

- **AMT**: Embedded ПО для сбора данных с газовых счетчиков, корректоров и обеспечения телеметрии этих данных; драйвера для GSM и WiFi модемов
- **CareFusion**: Портирование ПО для медицинского оборудования на новое железо: - Cortex-M4, и перенос ПО на новую embedded OS - SMX
- **Intel**: Написание моделей Intel Hardware для Simics симулятора
- **Samsung**: Разработка TizenRT OS: KASan, OTA, lock validator
- **Huawei**: ПО для базовой станции, немного компиляторы :)

Считаю System Software самой интересной областью в разработке ПО.
Каждый разработчик мечтает написать свой компилятор и свою ОС,
а мы этим занимаемся на работе !!!

Расскажите о вас кратко :)

- Какая область в разработке ПО вас интересует ?
- Чем бы вы хотели заниматься после окончания учебы в МФТИ ?
- Ваши Hard-Skills - что уже знаете и умеете ?

STM32F746G Discovery Kit:

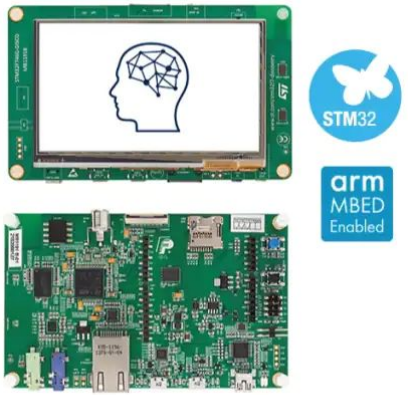
- ARM Cortex-M7 Core
- 1 MByte of flash (ROM), 340 KByte of RAM
- 480x272 LCD TFT, touch screen
- Ethernet
- USB
- 128 MBit SPI Flash Memory
- 128 MBit SDRAM
- 1 программируемая кнопка и 1 кнопка для сброса
- On-Board ST-LINK debugger
- Может быть запитана через USB

STM32F746G Discovery Kit, STM32CubeF7 Library:
https://github.com/STMicroelectronics/STM32CubeF7

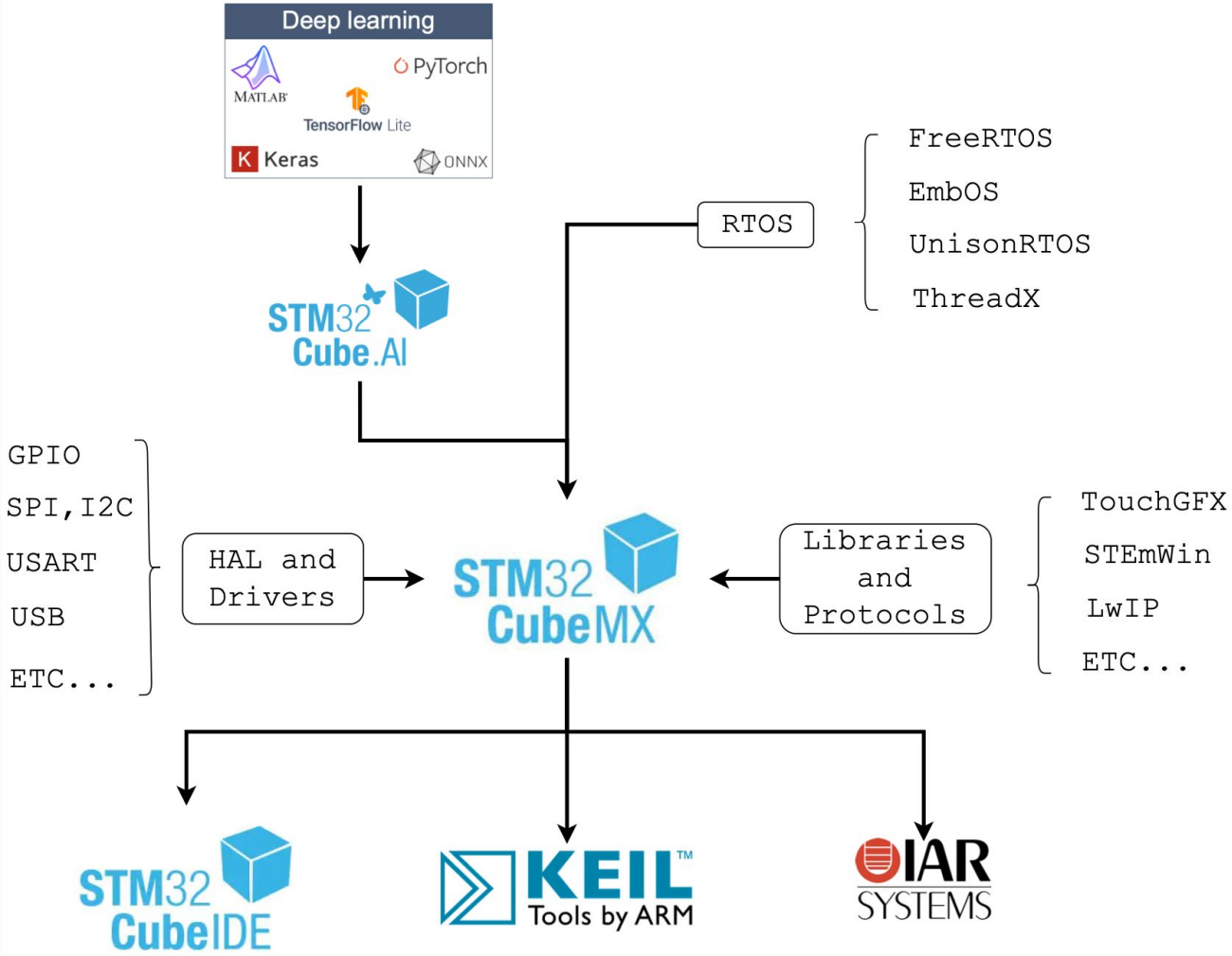
- *HAL (Hardware Abstraction Layer)*
- *TouchGFX graphics software stack*
- *USB, TCP-IP protocols*

IDE:

- *STM32CubeIDE:*
  https://www.st.com/en/development-tools/stm32cubeide.html

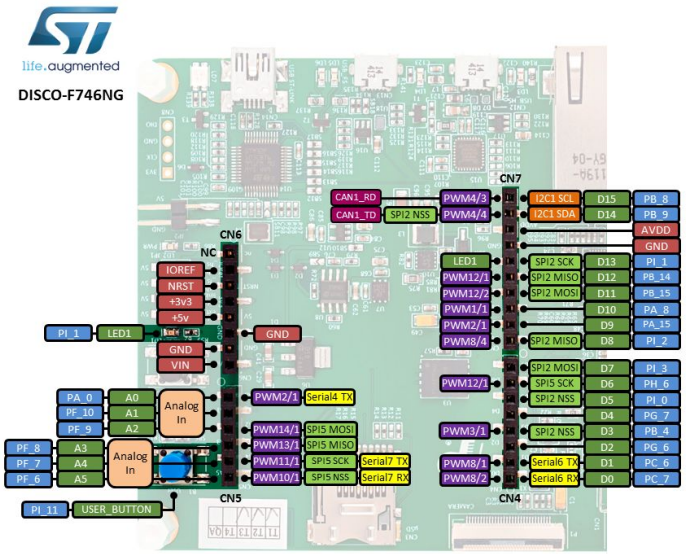


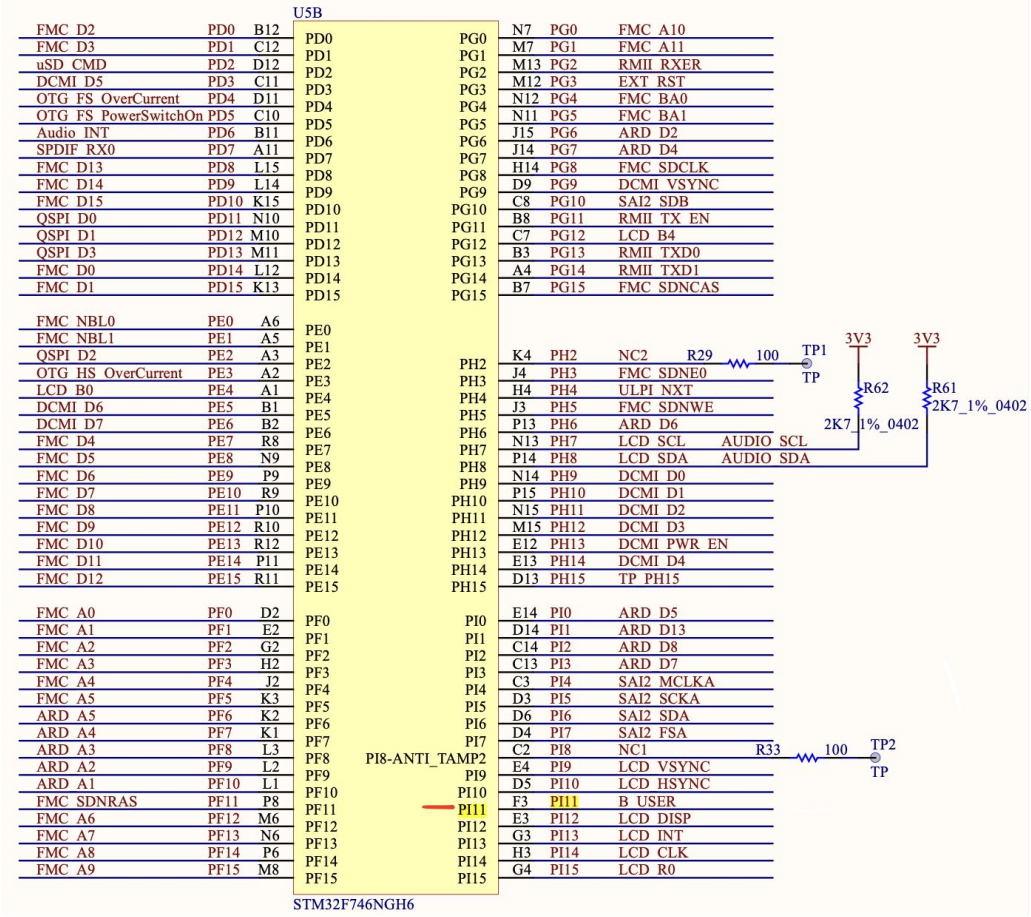
STM32F746G Discovery Kit

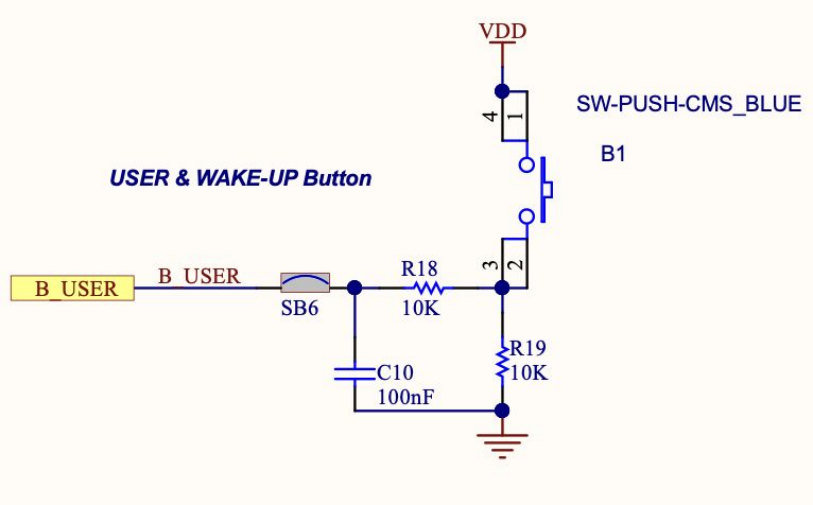

Deep learning
MATLAB
PyTorch
TensorFlow Lite
Keras
ONNX
STM32 Cube.AI
RTOS
FreeRTOS
EmbOS
UnisonRTOS
ThreadX
GPIO
SPI, I2C
USART
USB
ETC...
HAL and Drivers
STM32 CubeMX
Libraries and Protocols
TouchGFX
STEmWin
LwIP
ETC...
STM32 CubeIDE
KEIL Tools by ARM
IAR SYSTEMS

- https://drive.google.com/drive/folders/1EK-h3_woWxNoxjw-RlFXEqPOID82y8LI
  WMware Player Image - (Скачайте все 3 файла в одну папку для запуска)
- https://customerconnect.vmware.com/en/downloads/details?downloadGroup=WKST-PLAYER-1700&productId=1377&rPId=97014
  WMware Player - Free for personal use

- https://www.youtube.com/watch?v=PnDNqFKRsKI&t=224s
  Example: How to start, compile, debug with STM32Cube IDE







https://www.st.com/content/ccc/resource/technical/layouts_and_diagrams/schematic_pack/group1/ff/cd/ce/2d/f8/fb/40/69/mb1191-F746NGH6-C01_schematic/files/mb1191-F746NGH6-C01_schematic.pdf/jcr:content/translations/en.mb1191-F746NGH6-C01_schematic.pdf

```
diff --git a/Core/Src/main.c b/Core/Src/main.c
index c9f0c1a..8871a28 100644
--- a/Core/Src/main.c
+++ b/Core/Src/main.c
@@ -65,6 +65,9 @@ int main(void)
   /* Configure LED1 */
   BSP_LED_Init(LED1);

+  BSP_PB_Init(BUTTON_KEY, BUTTON_MODE_GPIO);
+  while (BSP_PB_GetState(BUTTON_KEY) != GPIO_PIN_SET) {};
+
   /*****************************************************/
```

6.4.1 GPIO port mode register (GPIOx_MODER) (x =A to K)

Address offset:0x00

Reset value:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	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]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODER[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

```
void BSP_PB_Init(Button_TypeDef Button, ButtonMode_TypeDef ButtonMode)
```

↓

```
void HAL_GPIO_Init(GPIO_TypeDef  *GPIOx, GPIO_InitTypeDef *GPIO_Init)
```

↓

```
GPIOI->PUPDR = GPIOI->PUPDR | (0b00 << 22); // << 22, потому что кнопка сидит на 11й ножке и 2 бита отвечаюта за каждую ножку (11*2=22)
GPIOI->MODER = GPIOI->MODER | (0b00 << 22);  // << 22, потому что кнопка сидит на 11й ножке и 2 бита отвечаюта за каждую ножку (11*2=22)
```

```
GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
  GPIO_PinState bitstatus;

  /* Check the parameters */
  assert_param(IS_GPIO_PIN(GPIO_Pin));

  if((GPIOx->IDR & GPIO_Pin) != (uint32_t)GPIO_PIN_RESET)
  {
    bitstatus = GPIO_PIN_SET;
  }
  else
  {
    bitstatus = GPIO_PIN_RESET;
  }
  return bitstatus;
}
```

Регистры микроконтроллера:
конфигурируют микроконтроллер (периферию микроконтроллера)

Периферия микроконтроллера:
Устройства в микроконтроллере которые не являются ядром микроконтроллера: GPIO, SPI, I2C, USART, ADC, DAC, etc…

NOTE:
HAL - Hardware Abstraction Layer
BSP - Board Support Package

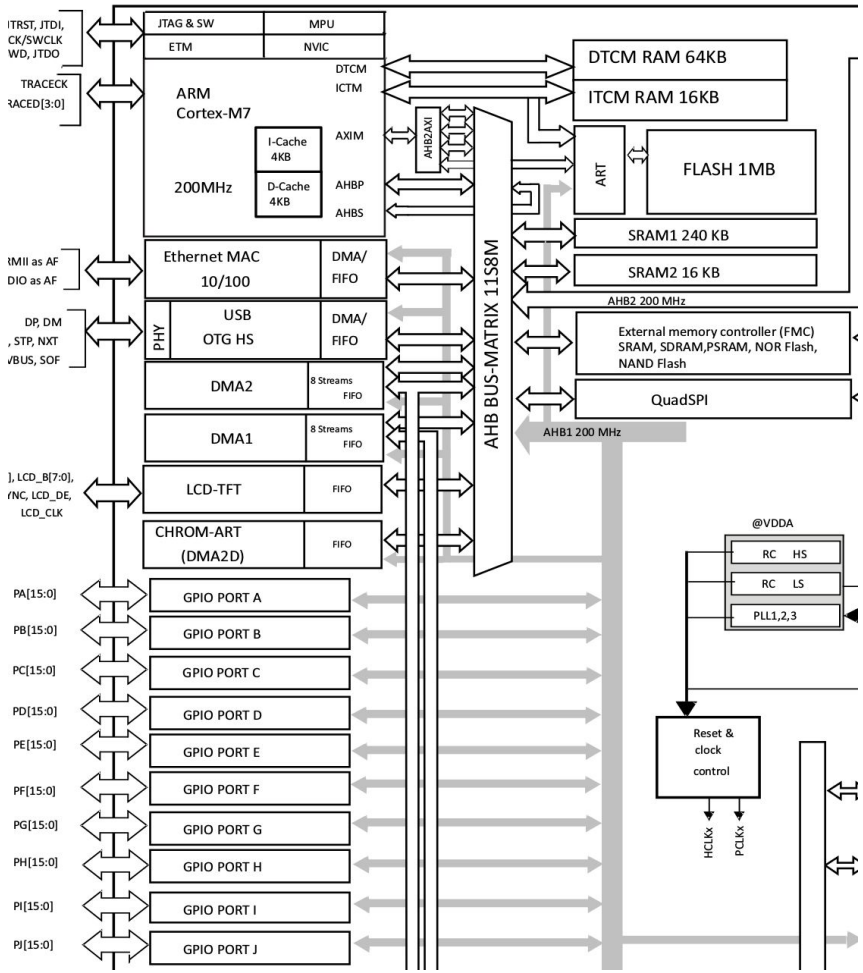
```
GPIOI->PUPDR = GPIOI->PUPDR | (0b00 << 22);
GPIOI->MODER = GPIOI->MODER | (0b00 << 22);
```

```
#define PERIPH_BASE             0x40000000UL
#define AHB1PERIPH_BASE         (PERIPH_BASE + 0x00020000UL)
#define GPIOI_BASE              (AHB1PERIPH_BASE + 0x2000UL)
#define GPIOI                   ((GPIO_TypeDef *) GPIOI_BASE)
```

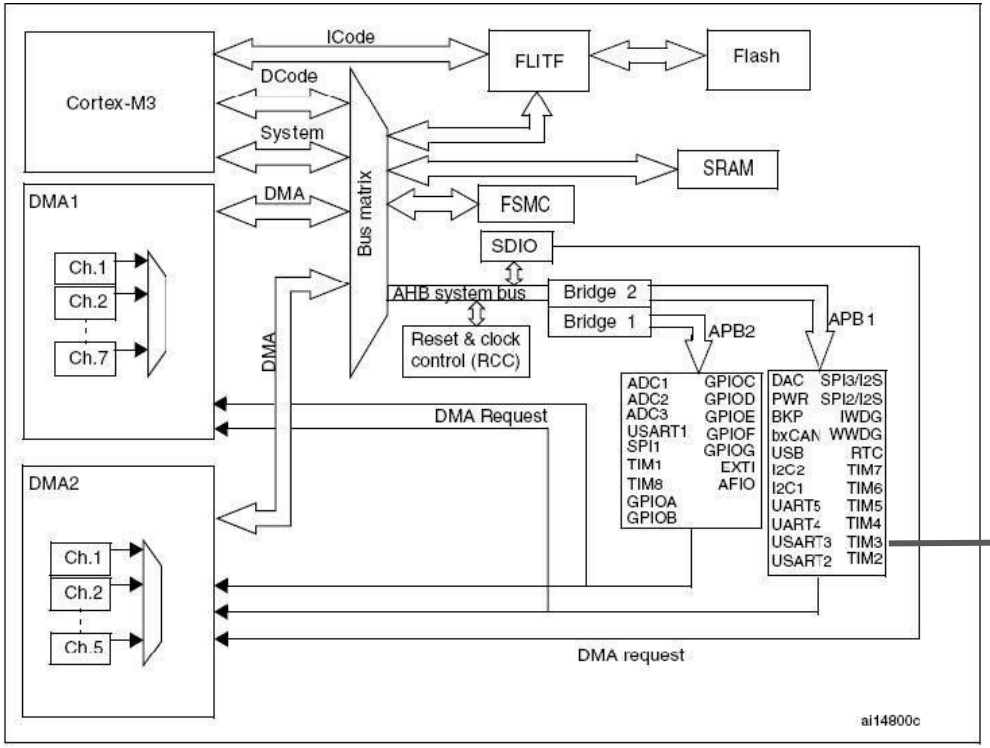
```
typedef struct
{
  __IO uint32_t MODER;    /*!< GPIO port mode register,               Address offset: 0x00      */
  __IO uint32_t OTYPER;   /*!< GPIO port output type register,        Address offset: 0x04      */
  __IO uint32_t OSPEEDR;  /*!< GPIO port output speed register,       Address offset: 0x08      */
  __IO uint32_t PUPDR;    /*!< GPIO port pull-up/pull-down register,  Address offset: 0x0C      */
  __IO uint32_t IDR;      /*!< GPIO port input data register,         Address offset: 0x10      */
  __IO uint32_t ODR;      /*!< GPIO port output data register,        Address offset: 0x14      */
  __IO uint32_t BSRR;     /*!< GPIO port bit set/reset register,      Address offset: 0x18      */
  __IO uint32_t LCKR;     /*!< GPIO port configuration lock register, Address offset: 0x1C      */
  __IO uint32_t AFR[2];   /*!< GPIO alternate function registers,     Address offset: 0x20-0x24 */
} GPIO_TypeDef;
```

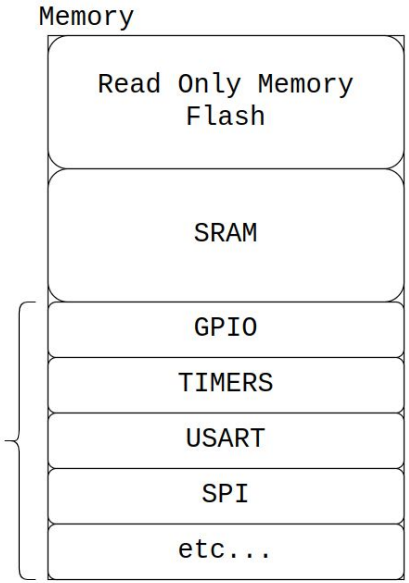
0x4002 2000 - 0x4002 23FF	GPIOI
0x4002 1C00 - 0x4002 1FFF	GPIOH
0x4002 1800 - 0x4002 1BFF	GPIOG
0x4002 1400 - 0x4002 17FF	GPIOF
0x4002 1000 - 0x4002 13FF	GPIOE
0x4002 0C00 - 0x4002 0FFF	GPIOD
0x4002 0800 - 0x4002 0BFF	GPIOC
0x4002 0400 - 0x4002 07FF	GPIOB
0x4002 0000 - 0x4002 03FF	GPIOA

Section 6.4.11: GPIO register map on page 214



https://www.st.com/resource/en/reference_manual/rm0385-stm32f75xxx-and-stm32f74xxx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf





Микроконтроллер = CPU Core + Peripherals

Что вынести из этой лекции:

- System and Embedded Software
- Регистры микроконтроллера позволяют вам конфигурировать периферию микроконтроллера и расположены по специальным адресам, которые описаны в https://www.st.com/resource/en/reference_manual/rm0385-stm32f75xxx-and-stm32f74xxx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf
- Microcontroller = CPU + Peripherals
- How to start new project in IDE, compile the code, flash the board and debug the code

Задание:

- Повторить пример - взять Hello World проект: https://www.youtube.com/watch?v=PnDNgFKRsKI&t=224s и добавить к нему логику управления кнопкой - 2 строчки на слайде предыдущем слайде (*Quick Demo слайд*). Скомпилировать, прошить в плату, принести на занятие.
- Найти и посмотреть в https://www.st.com/resource/en/reference_manual/rm0385-stm32f75xxx-and-stm32f74xxx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf описание следующих регистров:

```
GPIO->PUPDR
GPIO->MODER
GPIO->IDR
```

сделать скрин как на слайде *Quick Demo и прислать мне в подтверждение что вы посмотрели :)*

```
diff --git a/Core/Src/main.c b/Core/Src/main.c
index c9f0c1a..2abcdfd 100644
--- a/Core/Src/main.c
+++ b/Core/Src/main.c
@@ -19,6 +19,8 @@
 /* Includes ------------------------------------------------------------------*/
 #include "main.h"
 #include "WM.h"
+#include <stdio.h>
+#include "stm32746g_discovery.h"

 /* Private typedef -----------------------------------------------------------*/
 /* Private define ------------------------------------------------------------*/
@@ -27,6 +29,7 @@
 uint8_t GUI_Initialized = 0;
 TIM_HandleTypeDef TimHandle;
 uint32_t uwPrescalerValue = 0;
+volatile int button_cnt = 0;

 /* Private function prototypes -----------------------------------------------*/
 static void MPU_Config(void);
@@ -38,6 +41,11 @@ static void CPU_CACHE_Enable(void);

 /* Private functions ---------------------------------------------------------*/

+void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
+{
+  button_cnt++;
+}
+
 /**
   * @brief  Main program
   * @param  None
@@ -64,6 +72,7 @@ int main(void)

   /* Configure LED1 */
   BSP_LED_Init(LED1);
+  BSP_PB_Init(BUTTON_KEY, BUTTON_MODE_EXTI);

   /*****************************************************/

@@ -116,7 +125,16 @@ int main(void)
   MainTask();

   /* Infinite loop */
-  for(;;);
+  int last_button_cnt = 0;
+  char lcd_string[100] = {0};
+  for(;;) {
+    if (button_cnt != last_button_cnt) {
+      GUI_Clear();
+      sprintf(lcd_string, "Button cnt: %d", button_cnt);
+      GUI_DispStringAt(lcd_string, (LCD_GetXSize()-100)/2, (LCD_GetYSize()-20)/2);
+      last_button_cnt = button_cnt;
+    }
+  }
 }
```

```
diff --git a/Core/Src/stm32f7xx_it.c b/Core/Src/stm32f7xx_it.c
index 5107317..179b17c 100644
--- a/Core/Src/stm32f7xx_it.c
+++ b/Core/Src/stm32f7xx_it.c
@@ -156,3 +156,7 @@ void LTDC_IRQHandler(void)
   HAL_LTDC_IRQHandler(&hltdc);
 }

+void EXTI15_10_IRQHandler(void)
+{
+  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_11);
+}
diff --git a/STemWin/App/BASIC_HelloWorld.c b/STemWin/App/BASIC_HelloWorld.c
index e74c616..e06b665 100644
--- a/STemWin/App/BASIC_HelloWorld.c
+++ b/STemWin/App/BASIC_HelloWorld.c
@@ -36,7 +36,6 @@ void MainTask(void) {
   GUI_Clear();
   GUI_SetFont(&GUI_Font20_1);
   GUI_DispStringAt("Hello world!", (LCD_GetXSize()-100)/2, (LCD_GetYSize()-20)/2);
-  while(1);
 }
```