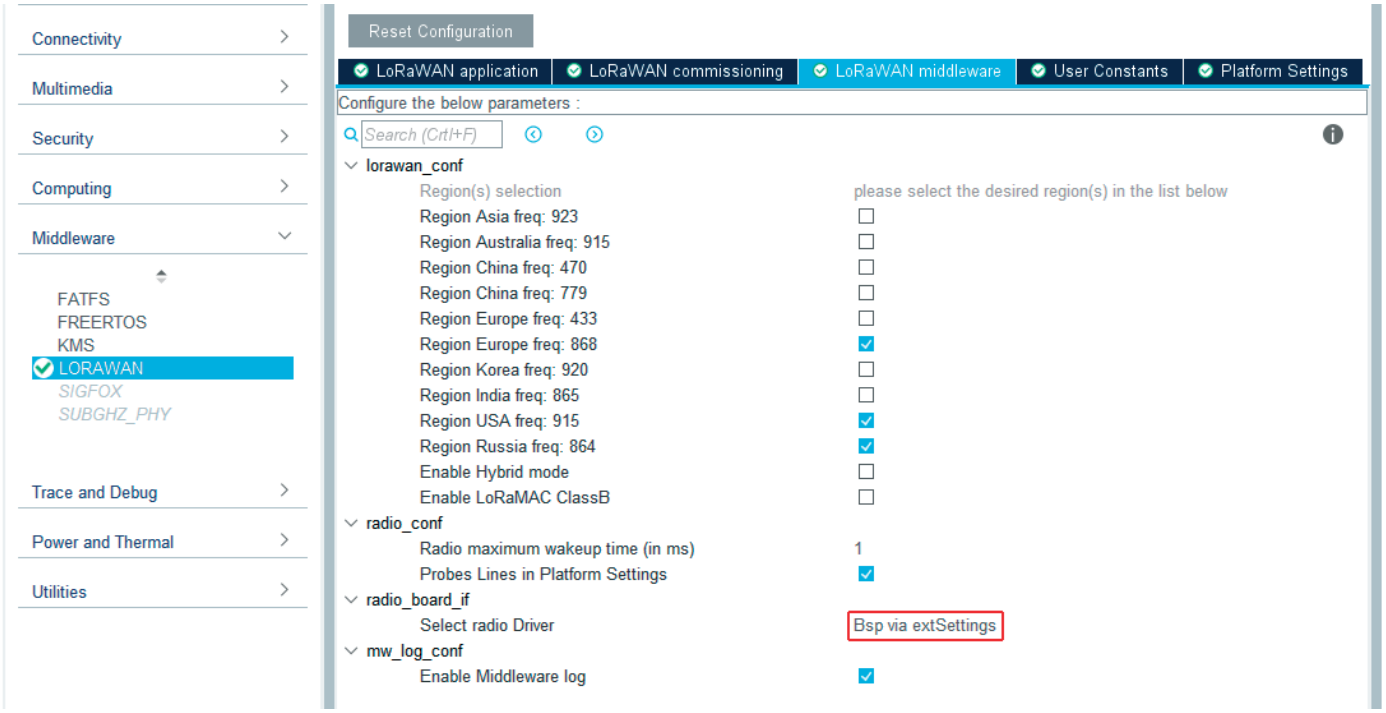
1. Открыл в CubeMX файл LoRaWAN_End_Node.ioc и сохранил его в новой папке. Добавил несколько кнопок и устройств.

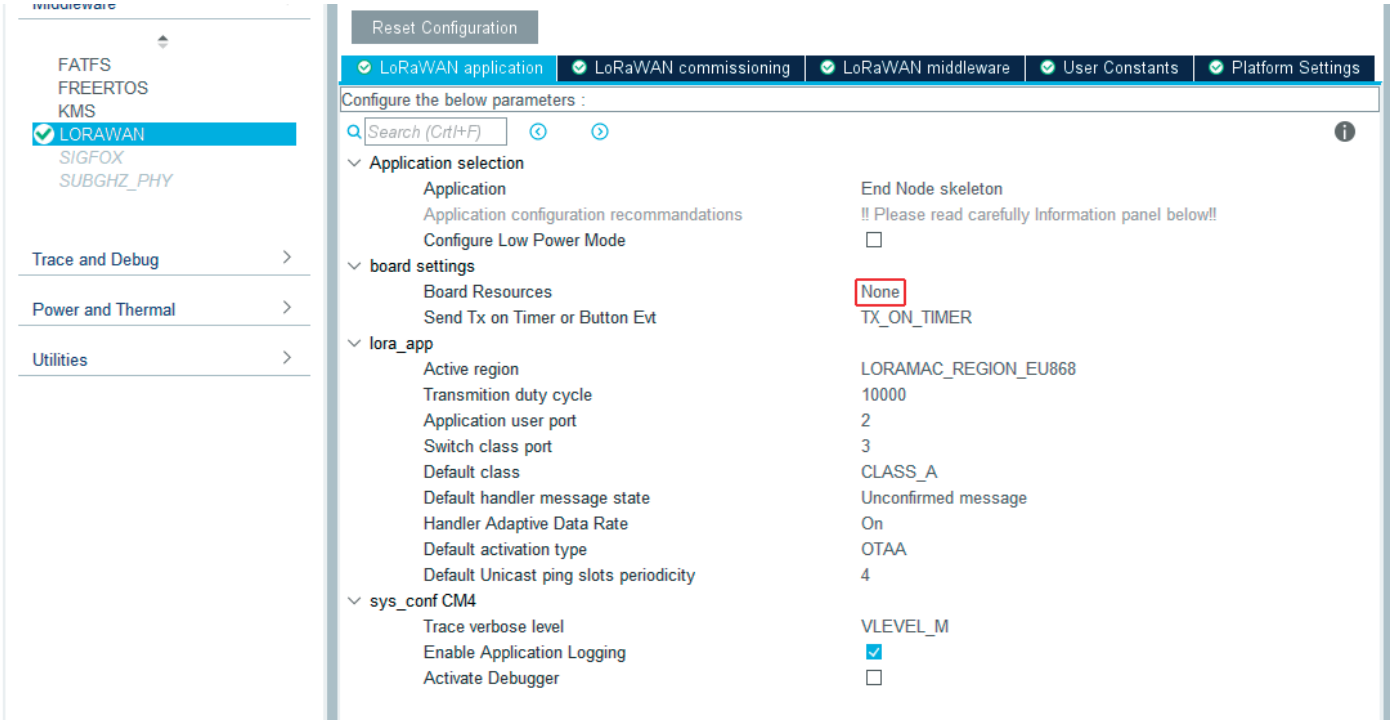
2. При генерации кода выдает предупреждение...



Убирается здесь!

user board плата пользователей

bsp via extSettings bsp через extSettings



3. Если не None, то в меню платформы добавляются кнопки и светодиоды. При этом создаются какие-то мутные файлы с описанием этих выводов схожие с описанием в BSP.

Не факт, что без этого заработают кнопки, но, забегая вперед, светодиоды моргают оптимистично.

Добавлять какие-либо свои выводы в CubeMX нужно через правую кнопку мыши. Во-первых он при этом не глючит с меткой пользователя. Во-вторых нуж-

но обязательно указывать принадлежность к ядру. Иначе метка вывода не попадает в файл main.h и вываливается ошибка. (Можно прям в файле поправить - добавить). Когда мы включаем интерфейсы, там сразу висит флажок ядра. С GPIO этого нет.

4. Генерируем код.

5. Проект не соберется, потому что нужно поправить файл lora_app.c

Это как-то пришло забавно... Ну выдает и выдает ошибки. Проект из библиотеки собирается, мой проект из CubeMX - нет. Начал сравнивать файлы. Нашел отличия в lora_app.c. Нужно из примера в библиотеке перетащить в наш файл все секции /* USER CODE BEGIN... */

В дальнейшем, прежде чем собирать проект, я сразу подсовывал ему отредактированный файл.

Примечание: В отредактированном файле изменены входной и выходной буферы. Чтобы входной буфер выводился, нужно поставить в sys_conf.h

```
#define VERBOSE_LEVEL  VLEVEL_H    // Высокий уровень трассировки log-a.
```

или сделать это до генерации кода.

Ну и

```
#define DEBUGGER_ENABLED    1      // Чтобы хоть как-то работал отлдчик
```

Можно поменять названия в файлах BSP, чтобы потом не путаться. Файл stm32wlxx_nucleo_conf_template.h, нужно скопировать в тот же каталог, но переименовать stm32wlxx_nucleo_conf.h без _template.

Примечание: Добавил сразу всего кучу: таймеры, SPI, DMA & e.t. Убил вечер. Так и не связался с Вега-сервером. Почему-то коннектиться начинает не по 168 порту, а по 240. Походу меняется класс устройства. Плюнул. Сегодня скопировал LoRaWAN_End_Node_01.ios в новую папку. Сгенерировал код, добавил BSP - заработало. Добавил 5 кнопок и провел генерацию кода в ту же папку. Работает. Переименовал библиотеку BSP - работает.

6. Читаем файл readme.txt примера. (Ну, наверное, это пунктом 1, но **только для двухядерного примера**).

- Настройте программное обеспечение через файлы конфигурации:

- CM0PLUS (конфиг Mw и радио драйверов)

- sys_conf.h, radio_conf.h, lorawan_conf.h, Commissioning.h, se-identity.h, mw_log_conf.h, main.h и т. д.

- CM4 (приложение Lora)

- sys_conf.h, lora_app.c, lora_app.h, nucleo_conf.h, main.h и т. д.

- Осторожно:

- регион и класс, выбранные в CM4 / LoRaWAN / App / lora_app.h, должны быть совместимы со списком CM0PLUS / LoRaWAN / Target / lorawan_conf.h

6.1. Размер сгенерированного файла lorawan_conf.h отличается от размера примера.

*/ * Экспортированные константы ----------*/*

*/ * Чтобы включить промежуточное ПО KMS с LoRaWAN, вы должны обновить эти файлы из примера проекта DualCore:*

- CM0PLUS / Core / Inc / kms_platf _objects_config.h: добавить все ключи LoRaWAN как структуры kms_object_keyhead_32_t.

- CM0PLUS / Core / Inc / kms_platf _objects_interface.h: добавить все ключевые индексы LoRaWAN

- CM0PLUS / Core / Inc / nvms_low_level.h: включить NVMS (энергонезависимую память) для хранения ключей сеанса

- CM0PLUS / Core / Src / nvms_low_level.c: реализация функций чтения / записи Flash для управления элементами NVMS.

И, наконец, измените определение LORAWAN_KMS на 1

Все честно правим.

Идем дальше по файлу lorawan_conf.h

```
/*!
 * Enables/Disables the context storage management storage.
 * Must be enabled for LoRaWAN 1.0.4 or later.
 */
#define CONTEXT_MANAGEMENT_ENABLED                  0
```

Это смущает.

Уже не смущает. Разработчики дописали его в следующей версии библиотеки.

Commissioning.h - требует версию

```
/*!
 * When using ABP activation the MAC layer must know in advance to which server
 * version it will be connected.
 */
#define ABP_ACTIVATION_LRWAN_VERSION_V10x           0x01000300 /* 1.0.3.0 */

#define ABP_ACTIVATION_LRWAN_VERSION                ABP_ACTIVATION_LRWAN_VERSION_
V10x

/*!
```

Тоже не до конца понятно.

7. Короче круть. все как в readme.txt примера

Запускаем два IAR одновременно. Второй загрузчик STLINK- V3 MINI не нужен, хотя руки чесались.

Зпускаем CM_4

Ставим точку останова на оператор, следующий за

HAL_PWREx_ReleaseCore(PWR_CORE_CPU2);

в файле sys_app.c

Он там и останавливается.

Запускаем CM0PLUS и он загружается

Начинаются фокусы

- Перестал регистрироваться на сервере по радио.

Ставим назад

#define LORAWAN_KMS 0
- Вега-сервер увидел.

Если включить

Выдает ошибку. Не находит файл nvmm.h

6. Правим библиотеку BSP.
Пусть она теперь называется STM32WLxx_Nucleo_Encoder
Открываем файл stm32wlxx_nucleo_Encoder.c
Начинаем с 42 строчки BUTTONn было 3, правим на 8.
И сначала stm32wlxx_nucleo_Encoder.h, как рука ляжет.

```
typedef enum
{
BUTTON_SW1       = 0,
. . . . .
Enc_DT           = 7,
}Button_TypeDef;
```

Все достаточно очевидно до
static uint32_t button_interrupt_priority[BUTTONn] = {BSP_BUTTON_SWx_IT_PRIORITY...
BSP_BUTTON_SWx_IT_PRIORITY лежит в файле stm32wlxx_nucleo_conf.h
Я не увидел необходимости менять приоритеты кнопок и плодить ссылки в других файлах.
Ну и пока не изведем все ошибки.

7. Создаем задачу реакции на кнопку.
Первое, что я нашел, инициализация

```
void LoRaWAN_Init(void)
{
  /* USER CODE BEGIN LoRaWAN_Init_1 */
  BSP_LED_Init(LED_BLUE);
  BSP_LED_Init(LED_GREEN);
  BSP_LED_Init(LED_RED);
  BSP_PB_Init(BUTTON_SW2, BUTTON_MODE_EXTI);
```

в файле lora_app.c.

Создадим группу на уровне Core Ext_Dev_App в группе USER.
Вставим туда файл ext.dev.app.c/
Мы в ручную модифицировали файл lora_app.c. В нем есть такая вставка

```
/* USER CODE BEGIN PB_Callbacks */
/* Note: Current the stm32wlxx_it.c generated by STM32CubeMX does not support BSP for PB in
EXTI mode. */
/* In order to get a push button IRS by code automatically generated */
/* HAL_GPIO_EXTI_Callback is today the only available possibility. */
/* Using HAL_GPIO_EXTI_Callback() shortcuts the BSP. */
```

```
/* If users wants to go through the BSP, stm32wlxx_it.c should be updated  */
/* in the USER CODE SESSION of the correspondent EXTIn_IRQHandler() */
/* to call the BSP_PB_IRQHandler() or the HAL_EXTI_IRQHandler(&H_EXTI_n);. */
/* Then the below HAL_GPIO_EXTI_Callback() can be replaced by BSP callback */
/* Примечание: текущий файл stm32wlxx_it.c, созданный STM32CubeMX, не поддерживает BSP для PB в режиме EXTI. */
/* Чтобы получить кнопку IRS по автоматически сгенерированному коду */
/* HAL_GPIO_EXTI_Callback сегодня единственная доступная возможность. */
/* Использование HAL_GPIO_EXTI_Callback () сокращает BSP. */
/* Если пользователи хотят пройти через BSP, необходимо обновить stm32wlxx_it.c */
/* в USER CODE SESSION соответствующего EXTIn_IRQHandler () */
/* для вызова BSP_PB_IRQHandler () или HAL_EXTI_IRQHandler (& H_EXTI_n) ;. */
/* Тогда приведенный ниже HAL_GPIO_EXTI_Callback () может быть заменен обратным вызовом BSP */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  switch (GPIO_Pin)
  {
    case  BUTTON_SW1_PIN:
      /* Note: when «EventType == TX_ON_TIMER» this GPIO is not initialized */
      /* Примечание: когда «EventType == TX_ON_TIMER» этот GPIO не инициализируется  */
         UTIL_SEQ_SetTask((1 << CFG_SEQ_Task_LoRaSendOnTxTimerOrButtonEvent), CFG_SEQ_Prio_0);
      break;
    case  BUTTON_SW2_PIN:
      break;
    case  BUTTON_SW3_PIN:
      break;
    default:
      break;
  }
}
/* USER CODE END PB_Callbacks */
```

Это переопределение слабой функции
__weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
в 515 строке файла stm32wlxx_hal_gpio.c

Подозреваю, что переопределять дважды не получится, придется держать это место в уме и прописать там наши кнопки.

Итак,
мы добавили заголовок события в файл utilities_def.h
модифицировали void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)

Адрес девайса читается из flsh в файле sys_app.c

6. Настраиваем приложение по книжке.
se-identity.h находится в папке.
\Projects\<target>\Applications\LoRaWAN\LoRaWAN_End_Node\LoRaWAN\App\se-identity.h
Или открываем ветку Middiewares -> LoRaWAN -> soft-se.c.
Там ключи

7. Запускаем Вега-сервер.
Запускаем IOT Vega Admin Tool V1.1.6_ru
Добавляем девайс.

Сервер Веги:
[RxDeferredPacket] Increase DR: constraint by preferDr value Dr5
Увеличение отложенного пакета Rx DR: ограничение предпочтительным значением Dr Dr5
[RxDeferredPacket] увеличить DR: ограничение на значение PreferDr Dr5

send now is confirmed via [xxx]. Waiting walidation
сейчас отправка подтверждена через [xxx]. Ожидание валидации