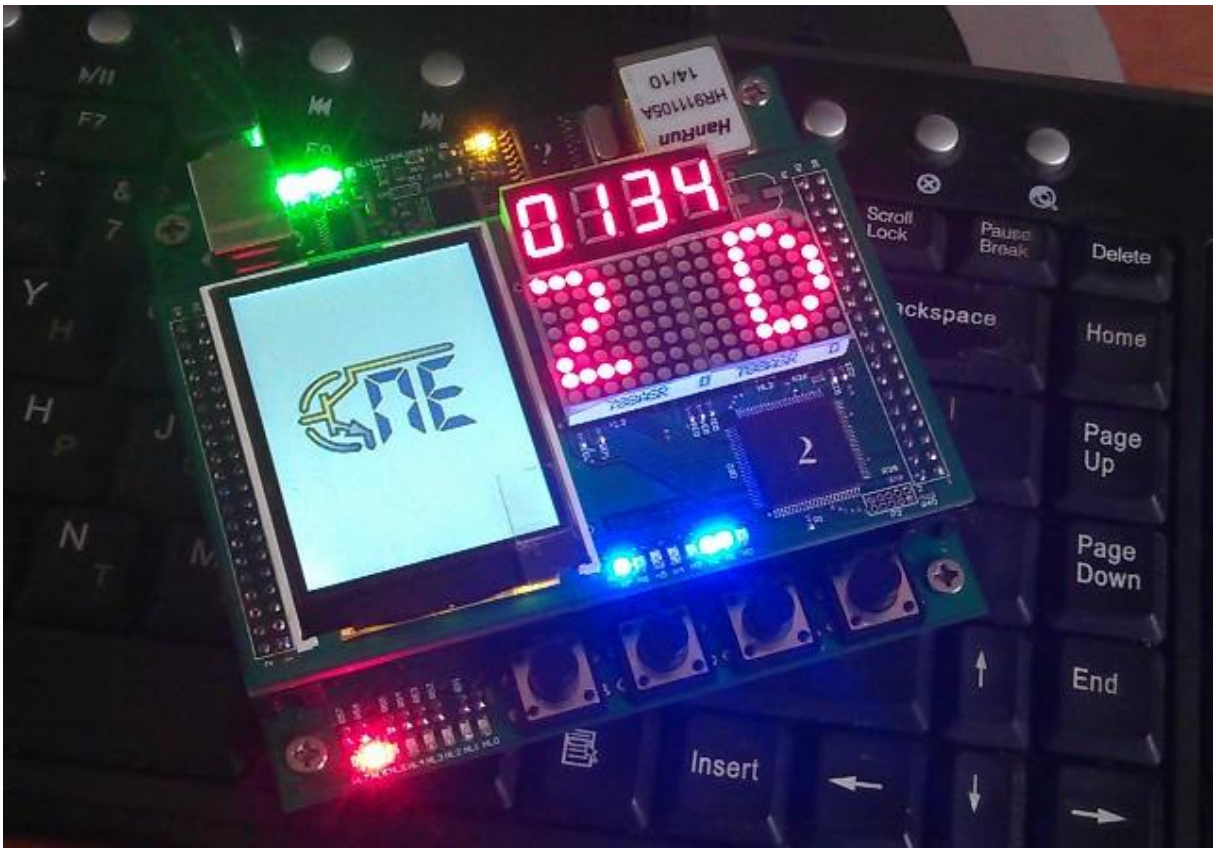
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРНІГІВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Навчально-науковий інститут технологій

МІКРОКОНТРОЛЕРИ STM32F4

МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ З ДИСЦИПЛІН «МІКРОПРОЦЕСОРНА ТЕХНІКА» ТА «ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ» ДЛЯ СТУДЕНТІВ СПЕЦІАЛЬНОСТЕЙ 171 «ЕЛЕКТРОНІКА» ТА 121 «ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»



Обговорено та рекомендовано на засіданні кафедри промислової електроніки.
Протокол № 6 від 27.01.2017 р.

ЧЕРНІГІВ – 2017

Мікроконтролери STM32F4. Методичні вказівки до виконання лабораторних робіт з дисциплін «Мікропроцесорна техніка» та «Програмне забезпечення спеціалізованих комп'ютерних систем» для студентів спеціальностей 171 «Електроніка» та 121 «Інженерія програмного забезпечення». – Чернігів: ЧНТУ, 2017. – 81 с.

Укладач: Войтенко Володимир Павлович, канд. техн. наук, доц.

Відповідальний за випуск: Денисов Юрій Олександрович, докт. техн. наук, проф., завідувач кафедри промислової електроніки

Рецензент: Ревко Анатолій Сергійович, канд. техн. наук, доц., доцент кафедри промислової електроніки Чернігівського національного технологічного університету

Зміст

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ІС – інтегрована схема.
МК – мікроконтролер.
МП – мікропроцесор (мікропроцесорний).
МПС – мікропроцесорна система.
ОЗП – оперативний запам‘ятовуючий пристрій.
ПДП (DMA) – прямий доступ до пам’яті (Direct Memory Request).
ПЗ – програмне забезпечення.
СКС – спеціалізована комп’ютерна система.
ЦАП – цифро-аналоговий перетворювач.
ЦОС (DSP) – цифрова обробка сигналу (Digital Signal Processing).
ШІМ (PWM) – широтно-імпульсна модуляція (Power Width Modulation).

ARM (ARM-МК) – процесор (мікроконтролер) архітектури ARM.
CMSIS – Cortex Microcontroller Software Interface Standard (стандарт програмного інтерфейсу мікроконтролерів Cortex)
FIFO – First-In, First-Out (стек з алгоритмом роботи пам’яті: перший увійшов, перший вийшов)
FPU – Floating Point Unit (співпроцесор з плаваючою точкою)
HAL – Hardware Abstraction Layer (шар апаратних абстракцій)
IoT – Internet of Things (Інтернет речей).
IDE – Integrated Development Environment (інтегроване середовище розробки програмного забезпечення).
MAC – Multiply with ACcumulate (множення з накопиченням).
MDK – Microcontroller Development Kit (набір програм Keil® для розробки ПЗ для МК від ARM Germany GmbH).
MPU – Memory Protection Unit (блок захисту пам’яті).
NMI – Non Masked Interrupt (немаскуєме переривання).
RTOS – Real Time Operation System (операційна система реального часу).
SIMD – Single Instruction, Multiple Data (множинні дані при одній команді)
SPL – Standard Peripheral Library

ВСТУП

ARM® Cortex®-M4 є вбудованим процесором високої продуктивності, який розроблений для застосувань, де потрібно простими засобами поєднати процедури керування з ефективною цифровою обробкою сигналу [1]. Процесор має багаті можливості налаштування конфігурації, що розширює діапазон реалізацій від тих, де потрібні операції з плаваючою крапкою, захист пам'яті та потужні технології трасування, до пристроїв, які чутливі до вартості та потребують мінімальних габаритів. На час написання даного навчального видання щонайменше десятеро відомих виробників ІС постачають на ринок мікроконтролери, які у своєму складі містять процесорне ядро *ARM® Cortex®-M4*. Номенклатура, сфери та обсяги застосування цих процесорів та різноманітних продуктів на їхній основі сягнули таких масштабів, що це вимагає від випускника технічного вишу хоча б первинного знайомства з особливостями їхньої архітектури та програмування *ARM® Cortex®-M4*.

Дані методичні вказівки призначені для самостійної підготовки студентів до виконання лабораторних робіт з дисциплін «Мікропроцесорна техніка», а також «Програмне забезпечення спеціалізованих комп'ютерних систем». Лабораторні роботи виконуються на спеціалізованому навчально-відлагоджувальному стенді *Inel-STM32* [2] і присвячені практичному дослідженню характерних особливостей мікропроцесорних систем та засобів відлагодження програмного забезпечення вбудованих процесорів *STM32F429ZGT6* на базі архітектури *ARM® Cortex®-M4*.

До складу кожного розділу входять основні відомості та деякі особливості підключення периферійних модулів у складі стенду. Процес підготовки до виконання роботи потребує обов'язкової самостійної роботи з рекомендованою літературою та лекційним матеріалом. Після теоретичної частини наведено список контрольних запитань, за допомогою яких оцінюється ступінь опанування теоретичного матеріалу і готовність студента до виконання лабораторної роботи. Далі наводиться перелік завдань, які потрібно вирішити в ході виконання роботи. Результати виконання робіт оформлюються у вигляді звітів, які мають містити таку інформацію:

1) короткі відомості про об'єкт вивчення (*15…20* рядків) з повною назвою, метою роботи і підписом студента, що виконав звіт, а також функціональну схему підключення певного периферійного модуля;

2) відповіді на контрольні запитання;

3) схему програми для МК і стислий опис алгоритму її дії;

4) текст програми для МК з коментарями;

5) висновки щодо отриманих результатів.

Пункти 1 – 3 оформлюються і показуються викладачеві на початку заняття, що є умовою допуску до виконання роботи в навчальній лабораторії. Весь звіт надається безпосередньо на захист роботи.

1 Лабораторна робота №1. Дослідження можливостей лабораторного стенду *Inel-STM* та особливостей ARM-МК

Мета роботи: вивчити побудову, можливості лабораторного стенду *Inel-STM*, ознайомитися з характеристиками та способами завантаження коду до ARM-МК.

1.1 Призначення і склад лабораторного стенду Inel-STM

Лабораторний стенд *Inel-STM* (рисунок 1.1) призначений для практичного ознайомлення з архітектурою вбудованого процесора *ARM® Cortex®-M4* та отримання навичок розробки програмного та апаратного забезпечення МПС та СКС на їхній основі. Крім того лабораторний стенд Inel-STM дозволяє вивчати, розробляти а також досліджувати системи керування, вимірювальні, охоронні, біомедичні та інші електронні системи із застосуванням різноманітних електронних компонентів, які можуть бути підключені до наявних інтерфейсів.

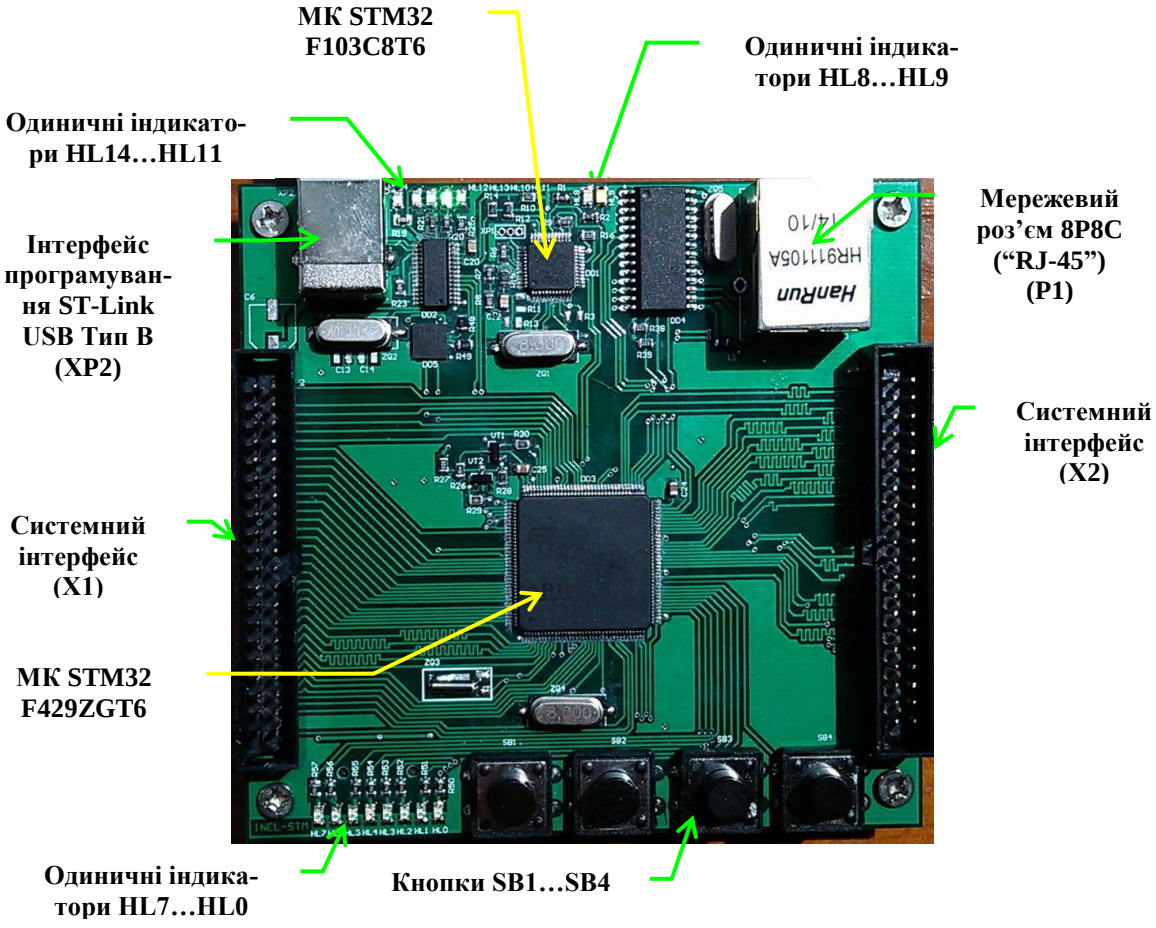


Рисунок 1.1 – Зовнішній вигляд лабораторного стенду *Inel-STM*

Для створення вхідного програмного модуля, його трансляції та відлагодження за допомогою крос-засобів використовується персональний комп‘ютер. Завантаження програми до резидентної пам'яті МК, а також подача живлення здійснюється через роз’єм USB XP2 за допомогою відповідного кабелю. Також у стенді реалізовано інтерфейс UART, який

пов’язується з персональним комп’ютером за допомогою USB-моста на основі ІС *Silab CP2102*.

Inel-STM являє собою мікрокомп’ютер з відкритою архітектурою, до системної шини якого підключені в якості зовнішніх пристроїв інші вузли та окремі ІС. Стенд містить наступні роз’єми розширення:

– USB-роз’єм Тип В (XP2) – послідовний інтерфейс ST-Link для зв‘язку з персональним комп‘ютером під час програмування, а також для живлення Inel-STM;

– мережевий роз’єм 8P8C (“RJ-45”) (P1) – послідовний порт для підключення Inel-STM до комп’ютерної мережі;

– два роз’єми системного інтерфейсу (X1 та X2), на які виведені частина ніжок МК *STM32F429ZGT6*, що дозволяє істотно змінювати функціональні можливості Inel-STM за рахунок підключення додаткових плат розширення.

Логіку завантаження резидентної пам’яті основного МК стенду Inel-STM *STM32F429ZGT6* реалізовано за допомогою допоміжного МК *STM32 F103C8T6* з 32-бітною архітектурою *ARM® Cortex®-M3*. Системний контролер керує режимами роботи, формує керуючі сигнали тощо.

Інтерфейс оператора у стенді Inel-STM утворюють наступні елементи:

– кнопки SB1…SB4;

– одиничні індикатори червоного кольору HL7…HL0 (User);

– одиничні індикатори жовтого кольору HL8…HL9 (COM);

– одиничні індикатори зеленого кольору HL14…HL11 (User).

На нижній стороні плати Inel-STM під одиничними індикаторами HL7…HL0 також розміщений роз’єм для підключення карт пам’яті стандарту MicroSD.

До системної шини стенду Inel-STM може бути під‘єднано одну з кількох модифікацій плат розширення. Базова плата розширення призначена для проведення лабораторних робіт, пов‘язаних з дослідженням систем відображення інформації.

1.2 Ключові особливості процесора ARM® Cortex®-M4

Процесор *ARM® Cortex®-M4* є модифікацією *ARM® Cortex®-M3* з наступними основними особливостями [1]:

1. Інтегрована підтримка цифрової обробки сигналу (*SIMD*, *MAC*) спрощує проектування системи, розробку та відлагодження програмного забезпечення, підсилює переваги МК.

2. Прискорення математичних операцій одинарної точності з плаваючою комою до 10 разів порівняно з еквівалентною цілочисельною програмною бібліотекою завдяки опціональному співпроцесору операцій з плаваючою крапкою (*FPU*).

3. Наявність великої кількості готових рішень для розмаїття ринків з повнофункціональним набором команд *ARMv7-M*.

4. Інтегровані режими очікування з програмним керуванням, гнучке керування тактуванням, опціональне утримання стану процесора. Це дає можливість ефективно використовувати енергію системи та досягнути виключної 32-розрядної продуктивності з низьким динамічним споживанням.

Cortex-M4 був спеціально розроблений з метою створення недорогих пристроїв високої продуктивності для широкого спектру сегментів ринку вбудованих засобів обробки цифрових сигналів, зокрема:

- автоматизація будівель, підприємств, установ та офісів;
- автомобільні та індустріальні системи керування;
- медичне устаткування;
- енергомережі, робототехніка, датчики;
- прилади для домашнього господарства, електронізований одяг;
- сільське господарство;
- торгівля, ідентифікація та відстеження, символіка;
- навколишнє середовище, розумні міста, розумне освітлення;
- розумні годинники;
- космос;
- VR-AR (віртуальна та доповнена реальність);
- Інтернет речей.

Спрощена структура базового процесора *ARM® Cortex®-M3* (рисунок 1.2) містить наступні вузли:

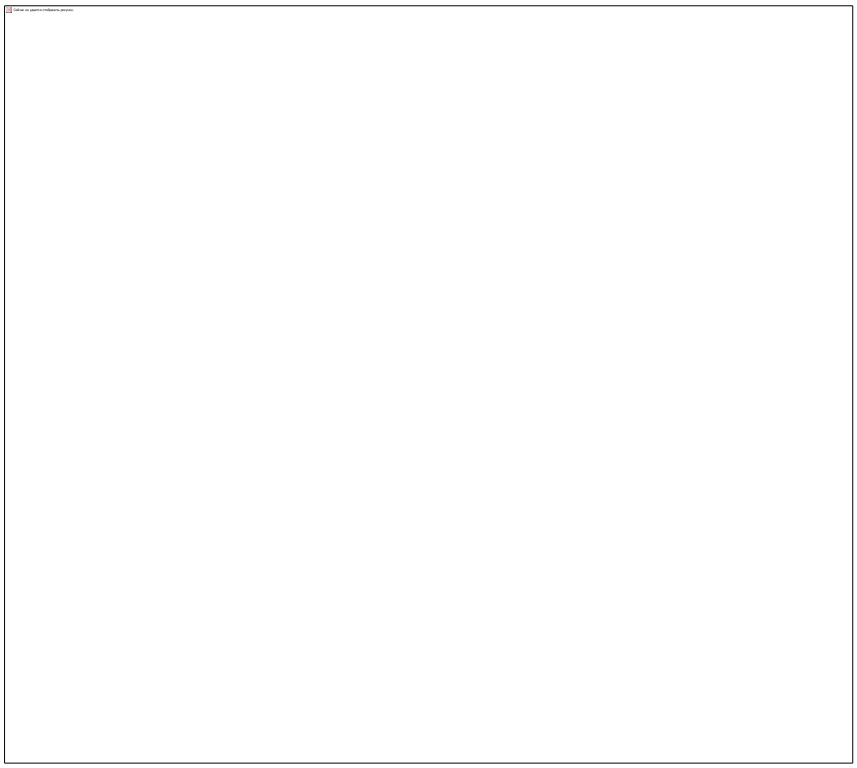


Рисунок 1.2 – Структура процесора *ARM® Cortex®-M4*

- *Nested Vectored Interrupt Controller Interface, NVIC* – контролер векторних вкладених переривань;
- *Wake Up Interrupt Controller, WIC* – інтерфейс контролера сторо-

жових переривань;

– *CPU/DSP* – ядро процесора *ARM® Cortex®-M4* з компонентами для цифрової обробки сигналу (ЦОС);

– *FPU, Floating Point Unit* – співпроцесор з плаваючою точкою;

– *Code interface* – інтерфейс коду програми;

– *Memory protection unit* – блок захисту пам'яті;

– *SRAM & Peripheral Interface* – інтерфейс статичного ОЗП та пристроїв введення/виведення.

– *Bus Matrix* – шинна матриця;

– *Data Watchpoint* –точки контролю даних;

– *Flash patch & Breakpoint* – трасування та корегування пам'яті;

– *ETM Trace, Embedded Trace Macrocell* – вбудована трасувальна макрокомірка);

– *DAP* – Debug Access Port (порт доступу до відлагодження);

– *Serial Wire Viewer* – контроль з однопровідним інтерфейсом.

Спрощена структура ядра базового процесора *ARM® Cortex®-M3* показана на рисунку 1.3.

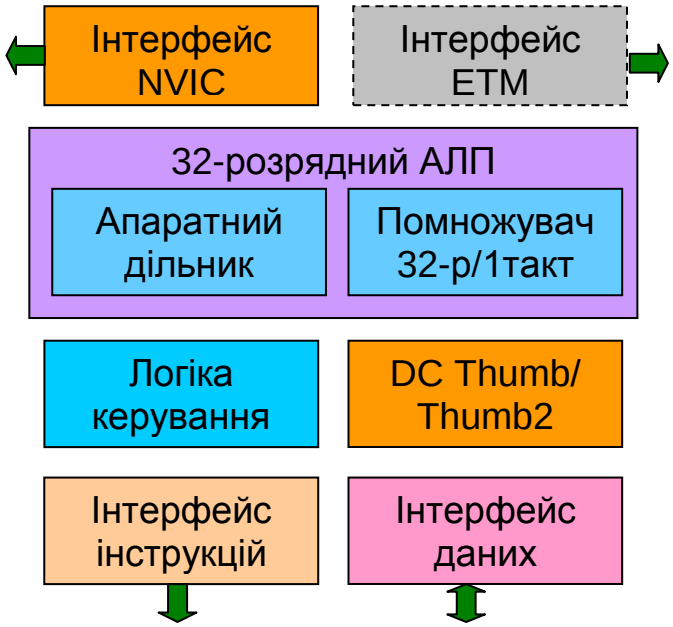


Рисунок 1.3 – Структура ядра *ARM® Cortex®-M3*

На час підготовки до публікації даного видання відомі наступні мікроконтролери на базі ядра *Cortex-M4*:

• *Atmel SAM4L, SAM4N, SAM4S, SAM4N*;

• *Freescale Kinetis K, W2*.

На базі ядра *Cortex-M4F* (*M4 + FPU*):

• *Atmel SAM4C (dual core), SAM4E, SAMG*;

• *Cypress Semiconductor FM4*;

• *Freescale Kinetis K, V3, V4*;

• *Infineon XMC4000*;

• *Nordic Semiconductor nRF52*;
• *NXP LPC4000, LPC4300* (один *Cortex-M4F* + один *Cortex-M0* ;
• *Silicon Labs / Energy Micro EFM32 Wonder*;
• *STMicroelectronics STM32 F3, F4, L4*;
• *Texas Instruments LM4F, TM4C, MSP432*;
• *Texas Instruments SimpleLink Wi-Fi CC3200* та *CC3200MOD*;
• *Toshiba TX04*.

Наступні ІС мають *Cortex-M4* або *M4F* в якості вторинного ядра:
• *Freescale Vybrid VF6* (один *Cortex-A5* + один *Cortex-M4F*);
• *Freescale i.MX6 SoloX* (один *Cortex-A9* + один *Cortex-M4F*);
• *Freescale i.MX 7 Solo / Dual* (1 або 2 *Cortex-A7* + один *Cortex-M4F*);
• *Texas Instruments OMAP 5* (1 *dual-core Cortex-A15* + два *Cortex-M4*).

1.3 Коротка характеристика МК STM32F429ZGT6

Концептуально *Cortex-M4* наслідує попередню модель (*Cortex-M3*), до якої долучена підтримка команд цифрової обробки сигналу. До ядра доданий співпроцесор з плаваючою точкою (*FPU*), тобто мова йде про *Cortex-M4F*.

Ключові особливості ядра *Cortex-M4* наступні.

- Ядро *ARMv7E-M* [3].
- Трьохступінчастий конвеєр із спекулятивними гілкуваннями.
- Набори команд:
  – *Thumb* (всі) та *Thumb-2* (всі);
  – 32-бітове знакове або без знаку апаратне множення з 32- або 64-розрядним результатом, додавання чи віднімання після множення;
  – 32-бітове апаратне ділення (2…12 циклів);
  – підтримка арифметики з насиченням;
  – розширення ЦОС: 16/32-бітне *MAC* за один цикл, подвійне 16-бітне *MAC* за один цикл, 8/16-бітна арифметика *SIMD*.
- Від 1 до 240 переривань, плюс *NMI*.
- Затримка при обробці переривань 12 циклів.
- Вбудовані режими сну.
- Блок *FPU* відповідає стандарту *IEEE-754* з одинарною точністю. Це називається розширенням *FPv4-SP*.
- Блок захисту пам'яті (*MPU*) до 8 областей.

МК *STM32F429ZGT6* характеризується наступними параметрами [4]:
1) резидентна флеш-пам'ять програм 1 *Мб*;
2) резидентна флеш-пам'ять даних 256 *Кб*;
3) тактова частота процесора – до 180 *МГц*;
4) інтерфейс камери та рідкокристалічного TFT-дисплея;
5) 114 портів загального призначення *GPIO*;
6) 3x12-розрядних АЦП, всього 24 канали; 2x12-розрядних ЦАП;
7) USB OTG HS/FS, Ethernet.

1.4 Генератор коду ініціалізації STM32CubeMX

STM32CubeMX є частиною оригінальної ініціативи *STMCube™*, яку виробник МК *STMicroelectronics* пропонує розробникам систем на базі *STM32* для зменшення зусиль, скорочення часу і витрат на власні проекти. *STM32CubeMX* впроваджує комплексну платформу програмного забезпечення, що відповідає серії (наприклад, *STM32CubeF4* для серії *STM32F4*). Ця платформа включає в себе *STM32Cube HAL* (вбудоване програмне забезпечення для *STM32* з рівнем абстракції, що забезпечує максимальну портативність), а також узгоджений набір компонентів проміжного ПЗ (*RTOS, USB, TCP/IP*, графіка). Всі вбудовані програмні утиліти поставляються з повним набором прикладів.

STM32CubeMX представляє собою графічний інструмент для налаштування програмного забезпечення, що дозволяє покроково генерувати відповідний код ініціалізації мовою С за допомогою графічних майстрів. Для початку роботи з генератором коду ініціалізації *STM32Cube* його треба скачати з [5] та інсталювати.

Робота з *STM32Cube* починається з вибору такого мікроконтролера *STMicroelectronics STM32*, який має необхідний набір периферійних пристроїв. Потім користувач повинен налаштувати вбудоване програмне забезпечення, застосовуючи *вирішувач конфліктів виводів*, *помічник установки дерева тактування*, *калькулятор енергоспоживання*, а також утиліту, що виконує *конфігурацію периферії* МК (*GPIO, USART*, ..) і стеків проміжного програмного забезпечення (*USB, TCP/IP*, ...).

Після запуску встановленого на комп'ютері *STM32CubeMX* необхідно додати бібліотеки під використовувані типи ядер МК. Це виконується на вкладці `Help->Install New Libraries`, де треба поставити галочку на необхідному пункті `Firmware Package For Family STM32F4` і натиснути кнопку `Install Now`. Після процесу скачування та розпаковування *STM32CubeMX* готовий до роботи. Також можна виконати інсталяцію бібліотек і в офлайн-режимі, обравши `From Local…` замість `Install Now`.

Далі в програмі *STM32CubeMX* треба обрати `New Project`. З меню, що випадає, знаходимо потрібну серію МК (`Series: STM32F4`), лінійку (`Lines: STM32F429/439`), корпус (`Package: LQFP144`). Далі у вікні списку `MK MCUs List` треба відібрати конкретну модель МК `STM32F429ZGTx` і натиснути `Ok`, що призведе до появи головного вікна конфігурування проекту (рисунок 1.4).

У цьому вікні розміщено декілька вкладок:

1) `Pinout` – виводи контролера, корпус якого представлений праворуч.
2) `Clock Configuration` – налаштування системи синхронізації.
3) `Configuration` – конфігурація проміжного ПЗ та периферії МК.
4) `Power Consumption Calculator` – калькулятор енергоспоживання.

Виходячи з того, яким чином використовуються ніжки («піни», *англ.* "pin") корпусу МК у стенді Inel-STM, зробимо відповідні налаштування. Задля цього послідовно будемо клацати мишкою на виводах портів МК та

з меню, що випадає, обирати функції портів відповідно до таблиці 1.1.

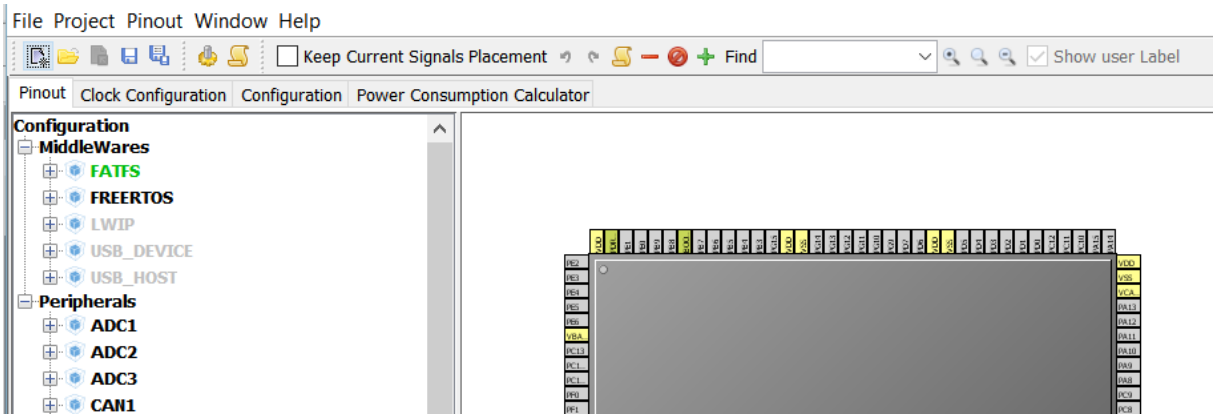


Рисунок 1.4 – Фрагмент вікна `Pinout` *STM32CubeMX*

Таблиця 1.1 – Функції, виконувані портами МК у стенді Inel-STM (мінімальні підключення)

Порт	Функція	Призначення
PA0	GPIO_Output	HL0
PA1	GPIO_Output	HL1
PA2	GPIO_Output	HL2
PA3	GPIO_Output	HL3
PA4	GPIO_Output	HL4
PA5	GPIO_Output	HL5
PA6	GPIO_Output	HL6
PA7	GPIO_Output	HL7

Порт	Функція	Призначення
PB0	GPIO_Input	SB3
PB1	GPIO_Input	SB4
PC4	GPIO_Input	SB1
PC5	GPIO_Input	SB2
PH0	RCC_OSC_IN	Резонатор 8 МГц
PH1	RCC_OSC_OUT	Резонатор 8 МГц

Ті порти МК, які не потрібно налаштовувати, залишаються у стані скидання і у вкладці `Pinout` *STM32CubeMX* мають сірий колір. Деякі виводи МК не підлягають конфігуруванню (живлення, скидання тощо) і мають забарвлення «хакі». В таблиці 1.1 ці виводи не відображені.

Для швидкого налаштування периферії МК, проміжного ПЗ та системи синхронізації можна скористуватись вікном з деревом периферії (ліворуч на рисунку 1.4). Так, для стенду Inel-STM зробимо наступне:

RCC (*Reset and Clock Control*, керування скиданням та тактуванням):

`High Speed Clock (HSE) -> Crystal/Ceramic Resonator`

УВАГА! Якщо неправильно проініціалізувати ніжки МК, відповідальні за послідовний інтерфейс відлагодження, або налаштувати ці ніжки на виведення, можна унеможливити використання стандартних засобів завантаження ПЗ (STM32 ST-LINK Utility). Виправлення такої ситуації потребує підключення ніжки МК BOOT0 до живлення (через резистор 1…7,5 кОм), а BOOT1 – до землі (також (через резистор). Далі за допомогою програми Flash Loader Demo, використовуючи UART1, треба очистити пам’ять МК.

1. Слід зазначити, що молодші вісім розрядів порту А МК (PA7…PA0) підключені безпосередньо до одиничних індикаторів червоного кольору HL7…HL0 (світлодіодів) на лабораторному стенді Inel-STM (Рисунок 1.1).

Саме тому ці ніжки МК маємо налаштувати, як виходи загального призначення (GPIO_Output).

2. З іншого боку, порти PC4, PC5, PB0, PB1 підключені до кнопок SB1…SB4 на лабораторному стенді Inel-STM (Рисунок 1.1), а отже, – мають бути входами загального призначення (GPIO_Input).

Конфігурування лінії порту на введення або виведення відмічається зеленим кольором відповідної ніжки МК і може потребувати додаткових налаштувань на вкладці Configuration -> System -> GPIO.

Після подібних налаштувань ми зможемо спостерігати, як деякі ніжки МК стануть зеленими (наприклад, як на рисунку 1.5), тобто буде не тільки визначено альтернативну функцію портів, але й одночасно обраний відповідний режим резидентної периферії. Проте також будуть з'являться попереджувальні позначки у вигляді жовтих трикутників і червоних кружечків навколо інших периферійних пристроїв у списку Configuration головного вікна *STM32CubeMX*. Це дозволяє уникнути конфліктів виводів.

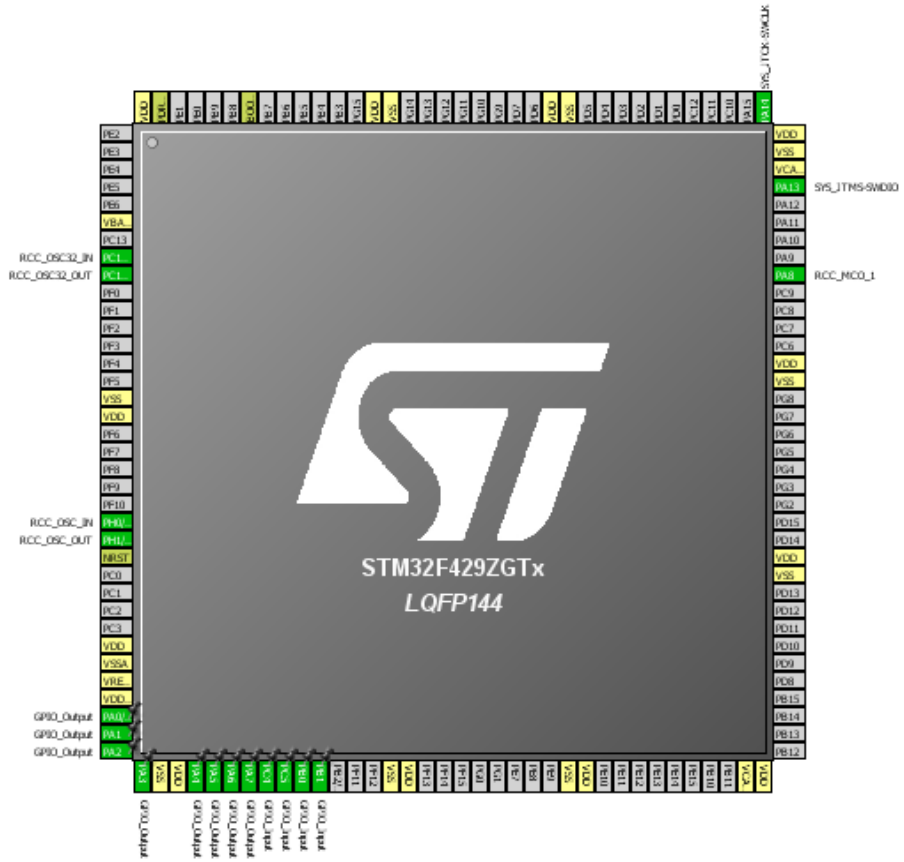


Рисунок 1.5 – Відображення виводів МК після налаштування

Оранжевий колір ніжки та позначка «no mode» вказує на відсутність налаштування певного периферійного пристрою у складі МК. Вибір режимів периферії автоматично встановлює як вивід МК, так і режими за замовчуванням для периферійних пристроїв, доступних у МК. Це означає, що *STM32CubeMX* буде генерувати С-код ініціалізації для всіх доступних периферійних пристроїв, а не тільки для тих, що мають відношення до додатка користувача. На це треба звернути увагу з погляду не тільки надлиш-

ковості (громіздкості) коду, але й можливості нічим не обумовленого зростання енергоспоживання внаслідок подання синхросигналів на ті периферійні пристрої, використання яких не планується.

Далі треба перейти до вкладки `Clock Configuration`, де вручну доведеться виставити частоту застосовуваного у стенді Inel-STM кварцового резонатора, задавши `Input frequency=8MHz`, а також коефіцієнт фазового автопідстроювання частоти `/M=8`. Зважаючи на використання зовнішнього джерела тактових сигналів, мультиплексор системи фазового автоматичного підстроювання частоти `PLL Source Mux` треба перемкнути у положення `HSE` (*High Speed External*). В подальшому під час використання годинника реального часу можна аналогічно «підключити» зовнішній кварцовий резонатор на частоту 32768 Гц для `LSE` (*Low Speed External*).

Далі ми маємо переконатися у правильності налаштування виводів (відсутності червоних маркерів) та підключити модулі проміжного ПЗ (ініціалізація, інтерфейси), перейшовши до вкладки `Configuration`.

Після того, як зроблені налаштування ніжок МК та резидентної периферії, ми впритул наблизилися до генерації коду ініціалізації засобами *STM32CubeMX*. Відкривши меню `Project->Settings…` (`Alt+P`), на вкладці `Project` вкажемо ім'я майбутнього проекту, наприклад, `My_First_F4`. Також пропишемо каталог розміщення (можна навіть створити новий, але ніде не застосовуйте кирилиці). Обравши потрібне IDE, отримаємо щось подібне до показаного нижче (рисунок 1.6).

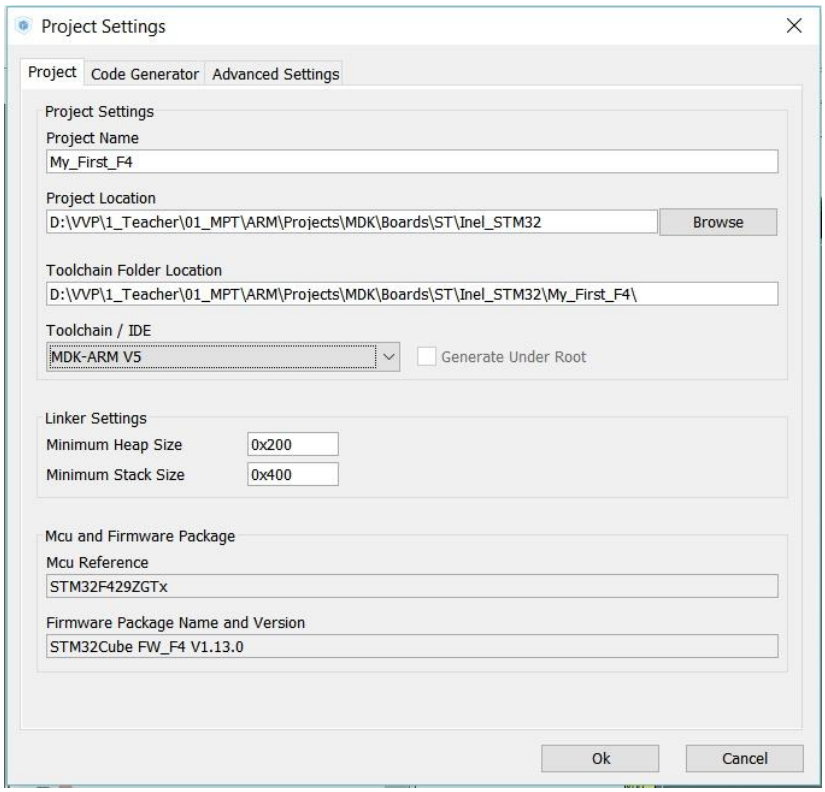


Рисунок 1.6 – Вкладка *Project->Settings STM32CubeMX*

Після вибору `Project-> Generate Code` та успішної генерації

стартового коду з'явиться запит про те, що ж робити далі (рисунок 1.7).

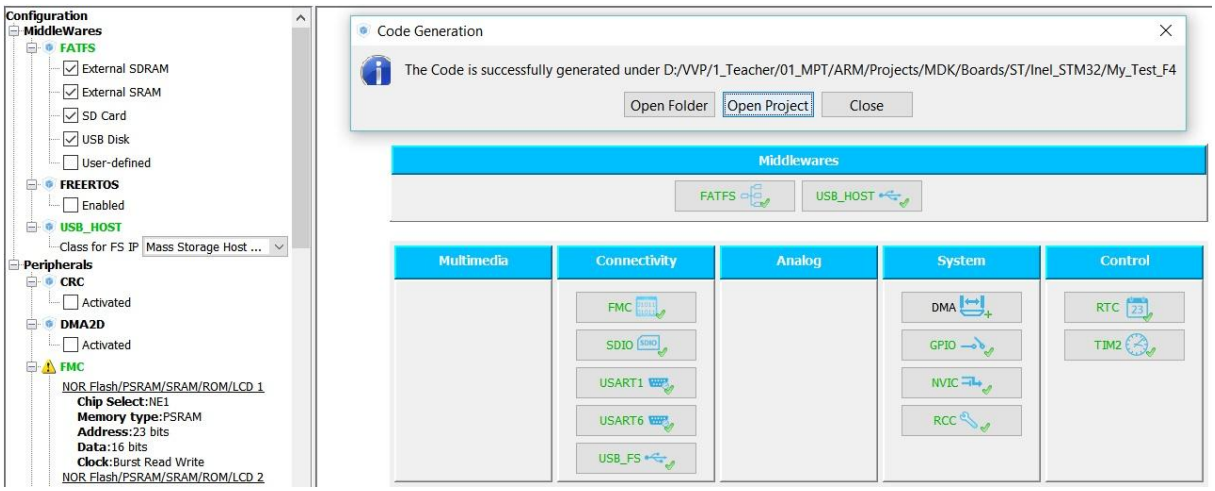


Рисунок 1.7 – Запит *STM32CubeMX* про подальші дії

На цьому етапі поки що завершимо роботу з *STM32CubeMX*.

1.5 Утиліта для програмування МК ST Microelectronics

Завантаження резидентної пам'яті МК можливе декількома способами. Зручною є безпосередня робота з інтегрованого середовища розробки програмного середовища, наприклад *IAR* або *Keil*. Крім того виробник МК *ST Microelectronics* пропонує безкоштовну утиліту *STM32 ST-LINK Utility*, яка дозволяє завантажувати файл відкомпільованої програми через програматор *ST-Link*, який поєднує інтерфейс *USB* персонального комп'ютера з послідовним інтерфейсом відлагодження *SWD*, що підтримується більшістю сучасних МК виробництва цієї фірми (сімейства *ST8*, *STM32*) [6].

Інтерфейс даної програми достатньо простий і дозволяє після під'єднання цільової плати з МК (Target->Connect Target) прочитати резидентну пам'ять МК, зберегти її вміст у двійковому або шістнадцятковому файлі або прочитати файл програми (File->Open File…) та завантажити його вміст до МК. Для отримання швидкого результату треба зробити:

Target->Settings…->Reset Mode->Core Reset;

Target->Program…->Download->Reset after programming->Start.

Inel-STM є, по суті, простим комп'ютером, який дозволяє виконувати програми, які підготовлені на персональному комп'ютері та завантажені у флеш-пам'ять програм МК *STM32F429ZGT6*. На даний час існує чимало інтегрованих середовищ розробки ПЗ та окремих компіляторів С, пристосованих до роботи з процесорами *ARM® Cortex®-M4*. Виробник *STM32F429ZGT6* – фірма *STMicroelectronics* – рекомендує 34 різновиди інструментального ПЗ для застосовуваного у стенді *Inel-STM* МК [4].

В даному виданні, а також у навчальних лабораторіях використовується пакети *Microcontroller Development Kit Keil® MDK*, а також *IAR Embedded Workbench*. Знайомство з першим з них розпочнеться з наступної лабораторної роботи.

1.6 Контрольні запитання

1. Наведіть призначення та склад лабораторного стенду *Inel-STM*.
2. Призначення та ключові особливості процесора *ARM® Cortex®-M4*.
3. Дайте коротку характеристику МК *STM32F429ZGT6*.
4. Опишіть, для чого призначений та охарактеризуйте можливості генератора коду ініціалізації *STM32Cube*.
5. Якими способами в генераторі коду ініціалізації *STM32Cube* можна додати бібліотеки під використовуваний тип ядра МК?
6. Опишіть, для чого призначена утиліта *STM32 ST-LINK Utility* та поясніть пункти меню цієї програми.

1.7 Хід роботи

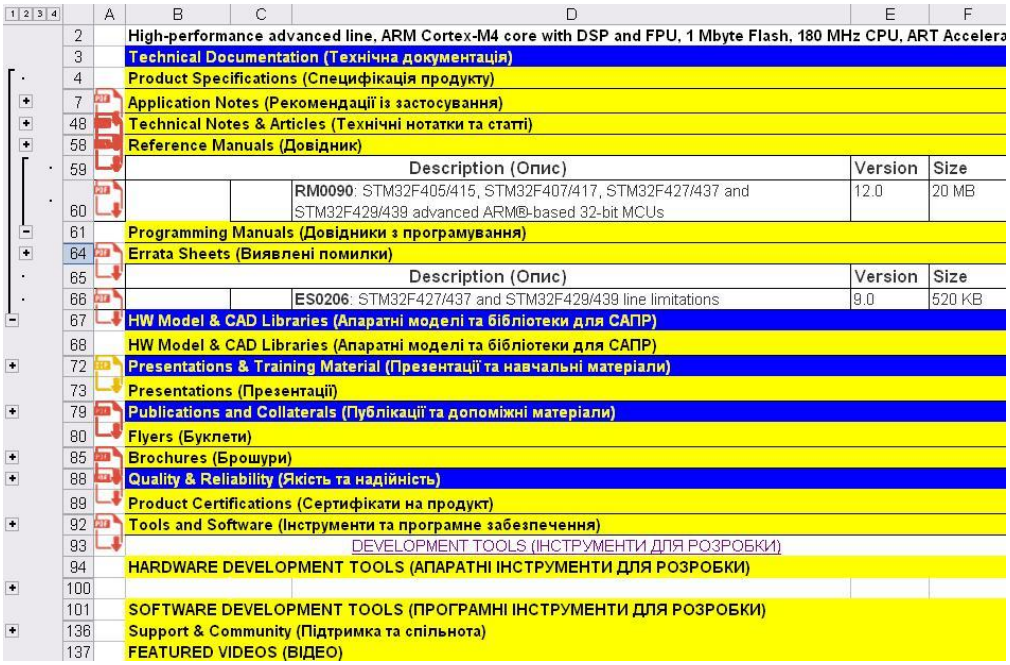
1.7.1 Підготовчі стадії

1. Ознайомитися із стендом Inel-STM та особливостями його складових частин (підрозділ 1.1).

2. Оформити першу частину звіту з виконання лабораторної роботи, в якій зазначити прізвища студентів бригади та дати письмові відповіді на контрольні запитання (п. 1.6).

3. Зареєструватися на офіційному веб-сайті *ST Microelectronics*, завантажити та встановити генератор коду ініціалізації *STM32Cube*.

4. На сторінці офіційного сайту *ST Microelectronics*, що присвячена МК *STM32F429ZG*, ознайомитися з наявною інформацією та скласти структуровану таблицю бази даних посилань на технічну документацію, додати переклад назв посилань. Приклад наведений на рисунку 1.8.



	A	B	C	D	E	F
2		High-performance advanced line, ARM Cortex-M4 core with DSP and FPU, 1 Mbyte Flash, 180 MHz CPU, ART Accelera				
3		Technical Documentation (Технічна документація)				
4		Product Specifications (Специфікація продукту)				
7		Application Notes (Рекомендації із застосування)				
48		Technical Notes & Articles (Технічні нотатки та статті)				
58		Reference Manuals (Довідник)				
59				Description (Опис)	Version	Size
60				RM0090: STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM®-based 32-bit MCUs	12.0	20 MB
61		Programming Manuals (Довідники з програмування)				
64		Errata Sheets (Виявлені помилки)				
65				Description (Опис)	Version	Size
66				ES0206: STM32F427/437 and STM32F429/439 line limitations	9.0	520 KB
67		HW Model & CAD Libraries (Апаратні моделі та бібліотеки для САПР)				
68		HW Model & CAD Libraries (Апаратні моделі та бібліотеки для САПР)				
72		Presentations & Training Material (Презентації та навчальні матеріали)				
73		Presentations (Презентації)				
79		Publications and Collaterals (Публікації та допоміжні матеріали)				
80		Flyers (Буклети)				
85		Brochures (Брошури)				
88		Quality & Reliability (Якість та надійність)				
89		Product Certifications (Сертифікати на продукт)				
92		Tools and Software (Інструменти та програмне забезпечення)				
93				DEVELOPMENT TOOLS (ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ)		
94		HARDWARE DEVELOPMENT TOOLS (АПАРАТНІ ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ)				
100						
101		SOFTWARE DEVELOPMENT TOOLS (ПРОГРАМНІ ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ)				
136		Support & Community (Підтримка та спільнота)				
137		FEATURED VIDEOS (ВІДЕО)				

Рисунок 1.8 – Фрагмент *Excel*-таблиці з посиланнями на інформаційні ресурси на МК *STM32F429ZG*

5. Закачати та ознайомитися, в першу чергу, з документами *RM0090* [7], *PM0214* [8].

1.7.2 Етап завантаження програмного забезпечення

1. Виконайте дії, описані в розділі 1.4.

2. Проаналізуйте автоматично створений *STM32Cube* програмний проект та нарисуйте схему програми, виділивши заголовки тих фрагментів, які призначені для вставлення коду користувача.

3. Користуючись підрозділом 1.5, завантажте тестову програму (файл `Test_Inel-STM.hex`) до стенду Inel-STM і переконайтеся у працездатності програмно-апаратного комплексу в цілому.

1.8 Вимоги до звіту по роботі

Звіт про виконання роботи повинен містити:

1. Титульний аркуш встановленого в університеті зразку.
2. Письмові відповіді на контрольні запитання.
3. Заповнену таблицю (аналог рисунку 1.8).
4. Схему програми автоматично створеного *STM32Cube* програмного проекту з функціональною схемою підключення застосованих індикаторів та кнопок.
5. Текст програми з перекладеними коментарями.
6. Висновки по роботі, в яких треба зазначити, чи досягнута мета. Що саме конкретно (перерахувати по пунктах) дозволяє досліджувати лабораторний стенд Inel-STM?

2 Лабораторна робота №2. Налаштування робочого місця для програмування ARM-МК в інтегрованому середовищі розробки програмного забезпечення Keil® MDK

Мета роботи: дослідити можливості Keil® MDK та ознайомитися з основами програмування ARM-МК, налагодженням і виконанням програм.

2.1 Короткі відомості про Keil® MDK

Microcontroller Development Kit Keil® MDK – одне з найбільш повних рішень з розробки програмного забезпечення для мікроконтролерів на основі ядра *ARM®*, що включає всі компоненти, які потрібні для створення, збирання та налагодження вбудованих програм [9].

Keil® MDK має модульну структуру і складається (рисунок 2.1) з інструментальної (*MDK-Tools*) та специфічної частин (*Software Packs*).

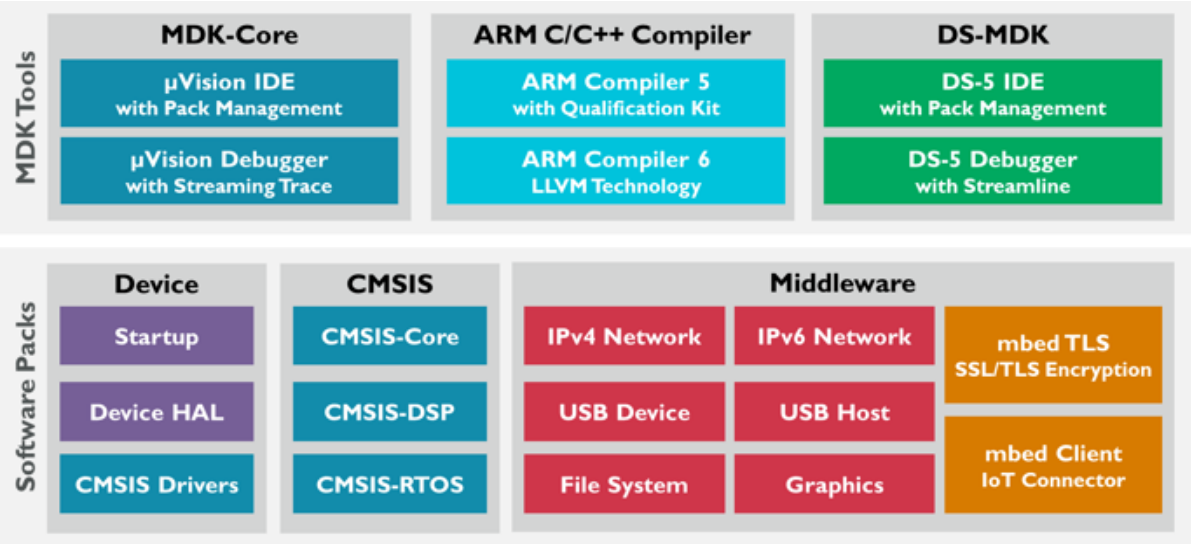


Рисунок 2.1 – Компоненти *Microcontroller Development Kit Keil® MDK*

Інструментальна частина, в свою чергу, містить ядро (*MDK-Core*), компілятор (*ARM C/C++ Compiler*), та відлагоджувальник (*DS-MDK*).

MDK-Core базується на IDE *µVision* з підтримкою пристроїв *Cortex-M*, включаючи нову архітектуру *ARMv8-M*.

Keil MDK підтримує два компілятори *ARM C/C++* з асемблером, компонувальником і бібліотеками, спеціально оптимізованими між розміром коду та продуктивністю під час виконання програми.

DS-MDK містить IDE/Відлагоджувальник *DS-5* на основі *Eclipse* і підтримує 32-розрядні процесори *Cortex-A* або гібридні системи з 32-бітними *Cortex-A* та *Cortex-M* (доступно з липня 2016 р.).

Software Packs (програмні пакети) містять компоненти підтримки МК і програмного забезпечення, які можна використовувати в якості будівельних блоків для розроблюваної програми. Програмні пакети можна додати до *MDK-Core* або *DS-MDK* в будь-який час, що забезпечить можли-

вість роботи з новими МК та оновлення проміжного програмного забезпечення незалежно від інструментального ПЗ. Пакети містять підтримку МК, бібліотеки *CMSIS*, проміжне програмне забезпечення, підтримку плат, шаблони коду, а також приклади проектів.

Блок *Startup*, зокрема, забезпечує формування коду ініціалізації МК, що суттєво спрощує початок програмування. Інші компоненти, які входять до *Software Packs*, заслуговують більш докладного вивчення по мірі просування роботи над проектом. Новим є стек мереж зв'язку *IPv4*/*IPv6*, який розширюється програмними компонентами *ARM mbed*™ для додатків Інтернету речей (*IoT*) [10].

MDK доступний в різних виданнях.

1. **MDK-Lite** містить повний набір для розробки програмного забезпечення для МК на основі *ARM*® і призначений для оцінки продукту, невеликих проектів, а також для ринку освітніх послуг. Підтримує мікроконтролери на основі *ARM Cortex*®*-M*, деякі процесори *ARM Cortex-R*, *Legacy ARM7*™, *ARM9*™ та пристрої на основі *ARM SecurCore*®. Має обмеження розміру коду до 32 *КБ*.

2. **MDK-Cortex-M** використовується для проектів, які засновані на мікроконтролерах *ARM Cortex-M*.

3. **MDK-Plus** застосовують для *Cortex-M*, *ARM7*, *ARM9*. Вміщує проміжне програмне забезпечення (мережі *IPv4*, пристрої USB, файлова система, графіка).

4. **MDK-Professional** призначений для *Cortex-M*, *Cortex-A*, *ARM7*, *ARM9*. Вміщує проміжне ПЗ (мережі *IPv4*/*IPv6*, *USB Host* та *Device*, файлова система, графіка, компоненти *mbed*).

2.2 Встановлення та дослідження Keil® MDK

Для того, щоб реалізувати в стенді Inel-STM свій перший проект для *STM32F429ZGT6*, необхідно зробити декілька кроків, які дозволять підготувати робоче місце та врешті-решт завантажити резидентну флеш-пам'ять цільового процесора шістнадцятковим файлом власної розробки.

1. По-перше, треба завантажити та встановити середовище відлагодження програмного забезпечення *MDK-Lite* [11]. На час написання цього навчального видання доступна версія `MDK521A.EXE` (619 262K).

Корисним буде дослідити директорію, де встановлене IDE (наприклад, – `D:\Keil_v5\`). У директорії D:\Keil_v5\UV4\ Ви знайдете власне *IDE µVision* (файл `UV4.exe`), а також іншу корисну програму – `PackInstaller.exe`, яка призначена для установки, оновлення і видалення програмних пакетів, і може бути запущена, як зсередини *IDE µVision*, так і автономно, поза *µVision*.

За умови автоматичного запуску `PackInstaller` на вкладці `Devices` треба розшукати актуальне сімейство та потрібний МК (див. нижче, п. 4 – у тому числі). Частина дій в наступних пунктах буде виконана автоматично.

2. По-друге, на офіційному сайті розробника *IDE* треба знайти сторінку підтримки цільового процесора. В даному випадку це `STMicroelectronics STM32F429ZGTx` [12]. На цій сторінці міститься пакет підтримки сімейства приладів *STMicroelectronics STM32F4*. На вкладці `Device Family Pack Support` є кнопка `Download`, натискання на яку призведе до завантаження поточної версії архівного файлу (наприклад, `Keil.STM32F4xx_DFP.2.9.0.pack`), що містить драйвери та корисні приклади програм для потрібного мікроконтролера.

На цій же сторінці можна побачити таблицю з переліком ресурсів, які можна використати під час розробки власних програм для мікроконтролера *STM32F429ZGT6*.

CMSIS розшифровується, як *Cortex Microcontroller Software Interface Standard* (стандарт інтерфейсу програмного забезпечення МК *Cortex*). Це незалежний від апаратних засобів рівень абстракції ПЗ для процесорів серії *Cortex-M*, який задає інтерфейси відлагодження [13]. Річ у тім, що створення програмного забезпечення є одним з основних факторів вартості в індустрії вбудованих систем. Завдяки стандартизації інтерфейсів програмного забезпечення для всіх постачальників ІС *Cortex-M* відбувається значне скорочення витрат, особливо при створенні нових проектів або міграції існуючого програмного забезпечення на новий пристрій.

CMSIS забезпечує цілісні і прості програмні інтерфейси процесора з периферійними пристроями, операційними системами реального часу, а також з проміжним програмним забезпеченням (*middleware*). Це спрощує повторне використання ПЗ, зменшуючи час навчання нових розробників мікроконтролерів і скорочуючи час виходу пристроїв на ринок.

Дослідіть уважно й інші ресурси на сторінці підтримки цільового процесора, адже це стане Вам у нагоді під час роботи над власними проектами, а також підготовки відповідей на контрольні запитання.

3. Завантажте та запустіть на виконання архівний файл пакету підтримки сімейства приладів *STMicroelectronics STM32F4* `Keil.STM32F4xx_DFP.2.9.0.pack`, і, через певний проміжок часу директорія `D:\Keil_v5\ARM\Pack\` суттєво зросте за обсягом. Уважно дослідіть склад директорії `D:\Keil_v5\ARM\Pack\Keil\STM32F4xx_DFP\ 2.9.0\`.

4. Після інсталяції продукту доцільно запустити допоміжну програму `D:\Keil_v5\UV4\PackInstaller.exe` (рисунок 2.2).

Рисунок 2.2 – Вікно привітання програми *PackInstaller.exe*

На вкладці `Devices` (рисунок 2.3) треба знайти цільовий мікроконтролер стенду Inel-STM, а саме, – *STM32F429* та виділити конкретний варіант виконання *STM32F429ZGTx*.

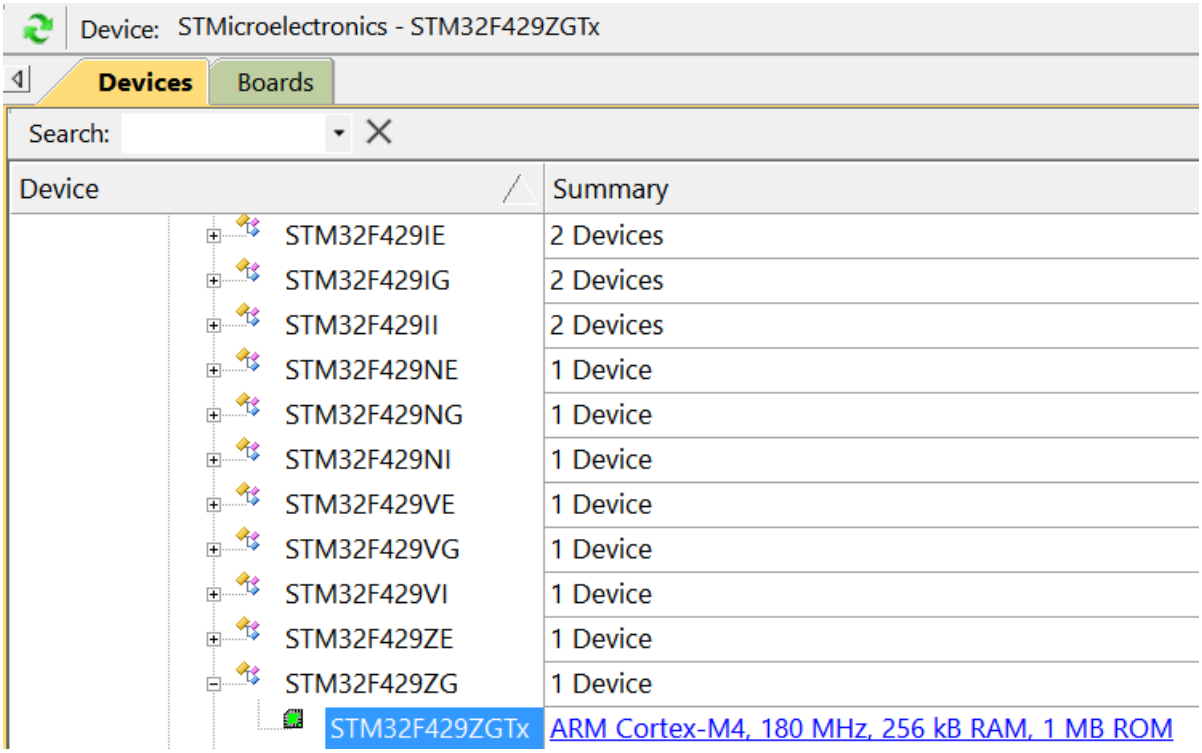


Рисунок 2.3 – Вкладка приладів, які підтримуються *Keil® MDK*

Якщо позначка потрібного приладу не має зеленого забарвлення, потрібно зробити налаштування, застосувавши вибір `Install` на вкладці `Packs` (рисунок 2.4).

У разі підключення до Інтернету на сайті `http://www.keil.com/` автоматично буде знайдений та встановлений пакет розширення для потрібного мікроконтролера [12].

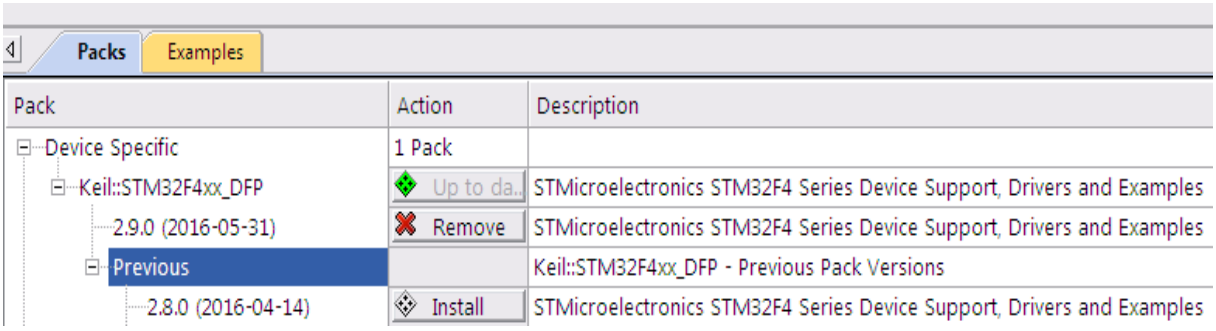


Рисунок 2.4 – Перевірка та завантаження пакету розширення мікроконтролера, який використовується в стенді

Зверніть увагу на корисну вкладку `Examples` (рисунок 2.5). Тут Ви побачите декілька фірмових програмних проектів для потрібного МК, які можна використати в якості основи власних розробок. Натискання на `Copy` дозволяє розархівувати певний проект у конкретне розташування на комп'ютері.

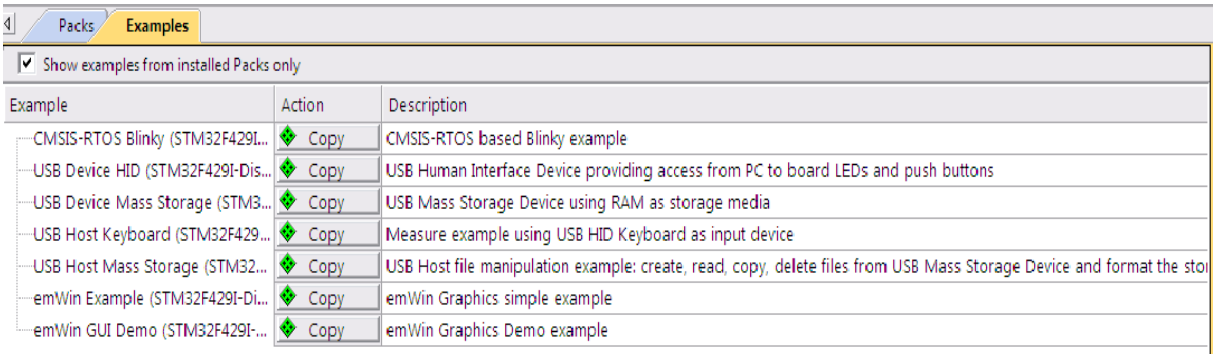


Рисунок 2.5 – Перегляд прикладів програмних проектів для мікроконтролера, який використовується в стенді

5. Скористаємося наданою можливістю засвоїти особливості розробки ПЗ на основі приклада і зробимо "*Copy*" прикладу під назвою:
`USB Device Mass Storage (STM32F429I-Discovery)`
`USB Mass Storage Device using RAM as storage media`

Цей приклад розроблений для іншої цільової плати (а саме, – *STM32F429I-Discovery*), а також для іншого мікроконтролера (*STM32F429I*). Проте є можливість використати даний проект і для стенду Inel-STM після певного доопрацювання.

6. У разі необхідності подальшої роботи з МК сімейства ARM7 також необхідно завантажити пакет підтримки цих контролерів на сторінці [14]. Для цього необхідно натиснути кнопку `Download Legacy support for ARM7, ARM9 & Cortex-R` (для останньої версії основної програми), або обрати потрібне посилання (для більш старих версій).

Результат інсталяції потрібних програм буде приблизно такий, як показано в таблиці 2.1.

Таблиця 2.1 – Потрібні завантаження

Назва пакету	Файл завантаження		Обсяг IDE після установки, ГБ
	Назва	Обсяг, МБ	
1. MDK Tools	MDK521a.exe	604	2,175
2. Software Packs	KeilSTM32F4xxDFP.29.0.pack	456	4,209
3. MDK ARM7 & ARM9 legacy support	MDK79521.exe	105	4,567

7. Після запуску `Keil µVision 5` з меню програм або з командного рядка (`D:\Keil_v5\UV4\UV4.exe`), виберемо в меню, що випадає:

`Project > Open Project…,`

а далі знайдемо безпосередньо у вікні та відкриємо потрібний нам файл проекту `MassStorage.uvprojx`

Примітка. Якщо першого разу відбувається автоматичний запуск `Keil µVision` після копіювання проекту за допомогою `PackInstaller.exe`, потрібний проект вже буде відкритий.

8. Обравши

`Project > Clean Targets,`

а далі:

`Project > Rebuild all target files,`

отримаємо вікно (рисунок 2.6).

Як ми бачимо, текст

`".\Output\MassStorage.axf" - 0 Error(s), 0 Warning(s).`

свідчить про те, що уся інструментальна система, а також усі файли фірмового проекту успішно взаємодіють.

9. Обов’язково дослідіть, які вкладки, крім `Project`, доступні у лівому вікні робочого простору `Keil µVision 5`.

10. По-третє, необхідно завантажити та встановити драйвер віртуального *COM*-порту IC мосту між *USB* та *UART CP210x* [15]. Цей драйвер потрібний для підтримки контактів хосту з продуктами на основі *CP210x*. Такі пристрої можуть також взаємодіяти з хостом, використовуючи драйвер прямого доступу *USBXpress*. Це нам знадобиться в наступних лабораторних роботах, де буде використовуватися зв'язок з комп’ютером через *COM*-порт.

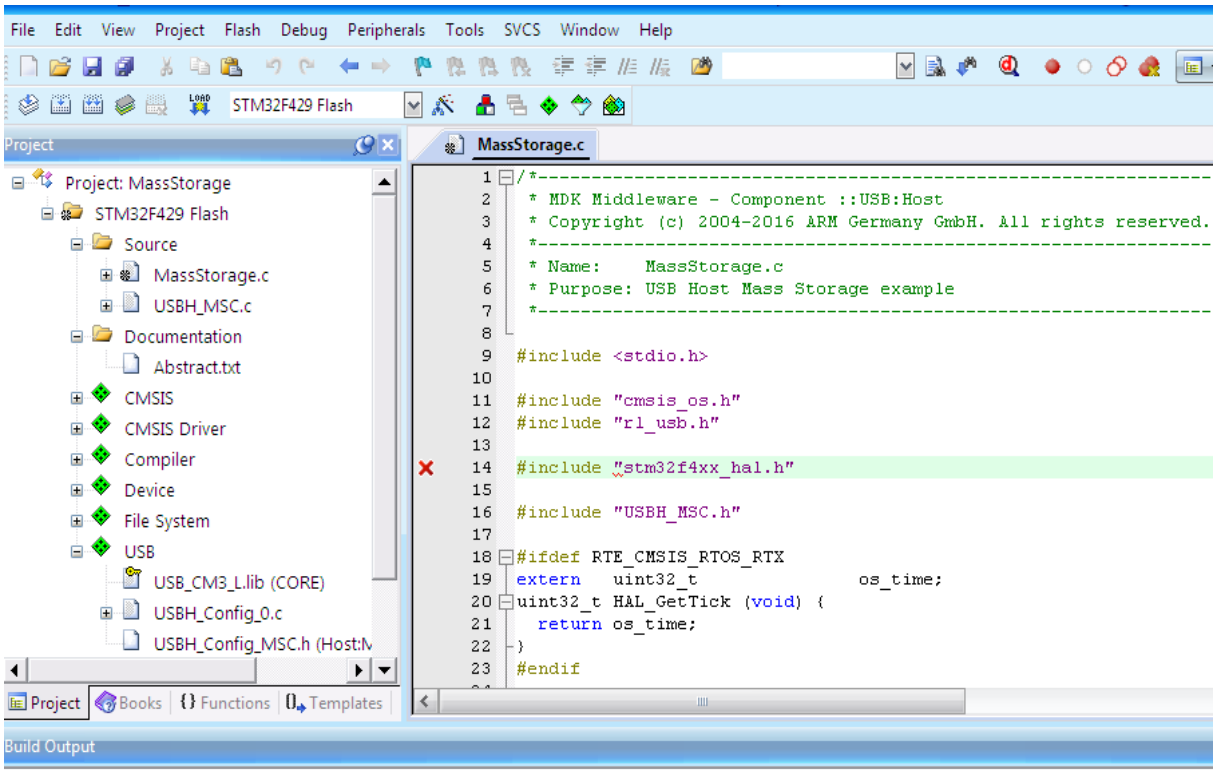


Рисунок 2.6 – Результат трансляції прикладу програмного проекту в *Keil® MDK*

11. Для подальшої роботи з *Keil® MDK* доцільно зробити деякі налаштування, обравши меню:

`Flash -> Configure Flash Tools… (Alt+F7).`

1.На вкладці `Device` можна довідатися про основні характеристики використаного МК (рисунок 2.7).

2.На вкладці `Target` треба встановити тактову частоту процесора таку, як і у стенді, тобто: `XTal (MHz) = 180`.

Примітка. Тут, здається, є неточність у позначенні параметра. Зазвичай `XTal` позначають частоту резонатора, а не тактування.

На вкладці `Output` треба дозволити створення шістнадцяткового файлу: `Create Hex file`. Цей файл може знадобитися у випадку завантаження МК іншою програмою, а не з *Keil® MDK*.

3.На вкладці треба обрати варіант відлагодження програми `Debug: Use Simulator` (стадія відлагодження коду без наявних апаратних ресурсів в режимі симулятора) або `ST-Link debugger` (відлагодження з реальним пристроєм).

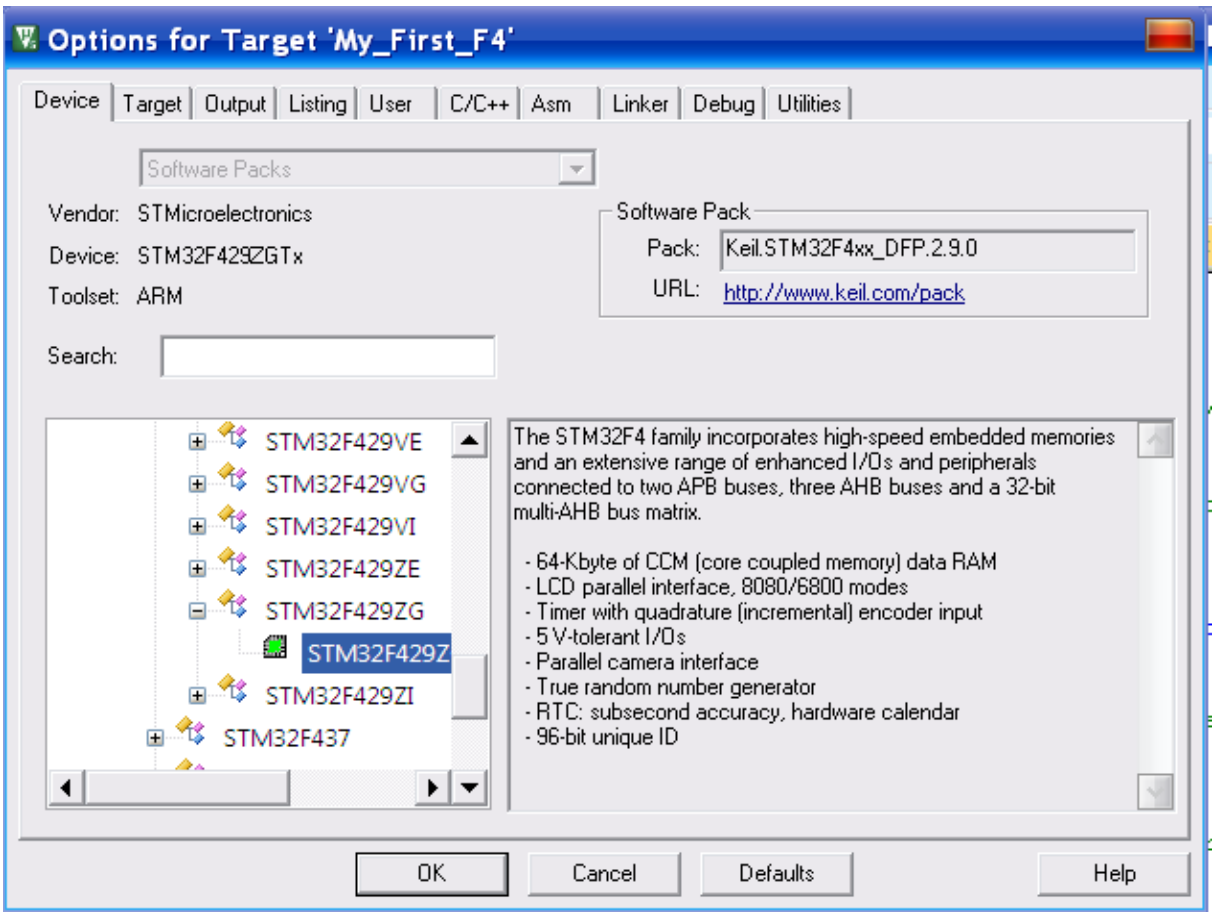


Рисунок 2.7 – Перевірка правильності вибору мікроконтролера

2.3 Створення коду ініціалізації для МК STM32F429ZGT6

Покроково розглянемо вирішення задачі програмування МК *STM32F429ZGT6* в середовищі *µVision*. Завдання полягає в наступному: після завантаження розробленої програми до МК мають засвітитися усі одиничні індикатори *HL7…HL0*. За натисканням кнопки *SB1 HL0…HL7* мають згаснути, а при повторному натисканні кнопки – знову засвітитися.

1. Розробимо схему програми (рисунок 2.8). У зв'язку з тим, що під час замикання механічного контакту (кнопки) можливе явище відскоку (*дзвону*, *bounce*), виникає перехідний процес з кількома послідовними замиканнями-розмиканнями. При цьому з лінії порту МК може бути зчитана випадкова послідовність нулів та одиниць. Для пригнічення цього небажаного явища в розроблюваній програмі застосовується найпростіший (але не самий ефективний) спосіб – затримка опитування контакту на час, який свідомо перевищує тривалість перехідного процесу дзвону контактів. Ця затримка підбирається експериментально для кожного типу датчика в межах 1…10 *мс*.

Крім того в програмі має бути реалізовано типовий інтерфейс з людиною-оператором, коли кнопку треба натиснути-відпустити для реалізації функції перемикання.

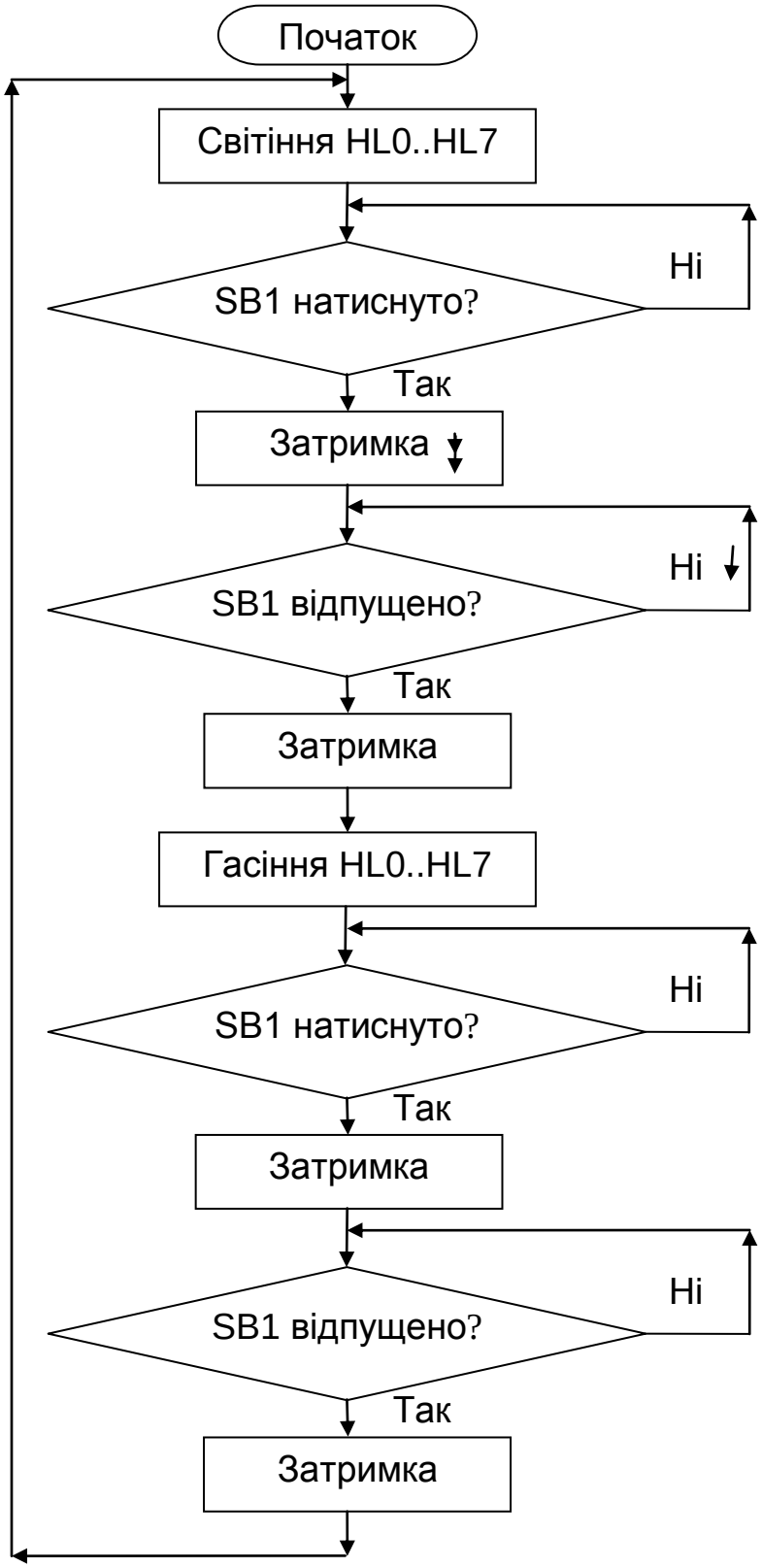


Рисунок 2.8 – Схема програми керування світлодіодами

2. Далі запускаємо програму *STM32CubeMX* (див. підрозділ 1.4). Натискаємо `New Project`, перевіряємо наявність оновлень (`Help/Check`

for Updates або Alt+C), знаходимо потрібний МК (STM32F429ZGTx) за допомогою фільтрів (за сімейством та лінією, корпусом), як було показано у підрозділі 1.4. Після натискання OK, зберігаємо проект (наприклад, з іменем Lab2.ioc) та на вкладці Pinout виконуємо такі ж самі налаштування, як і у Таблиці 1.1.

3. Після налаштування портів перейдемо до вкладки з деревом тактування МК (Clock Configuration). Задамо частоту зовнішнього високочастотного резонатора Input Frequency **8 MHz**. За допомогою установки коефіцієнтів і вибору входів мультиплексорів доб'ємося максимальної частоти роботи ядра МК у 180 *МГц*:

```
PLL Source Mux = HSE;
PLLM = 4;
PLLN = 180;
PLLP = 2;
System Clock Mux = PLLCLK;
SYSCLK (MHz) = 180.
```

Червоні підсвічування певних блоків свідчать про перевищення максимальної частоти тактування конкретного периферійного пристрою. Збільшивши коефіцієнти дільників частоти (APB1 Prescaler = 4; **APB2 Prescaler = 2**), отримаємо адекватні налаштування (рисунок 2.9).

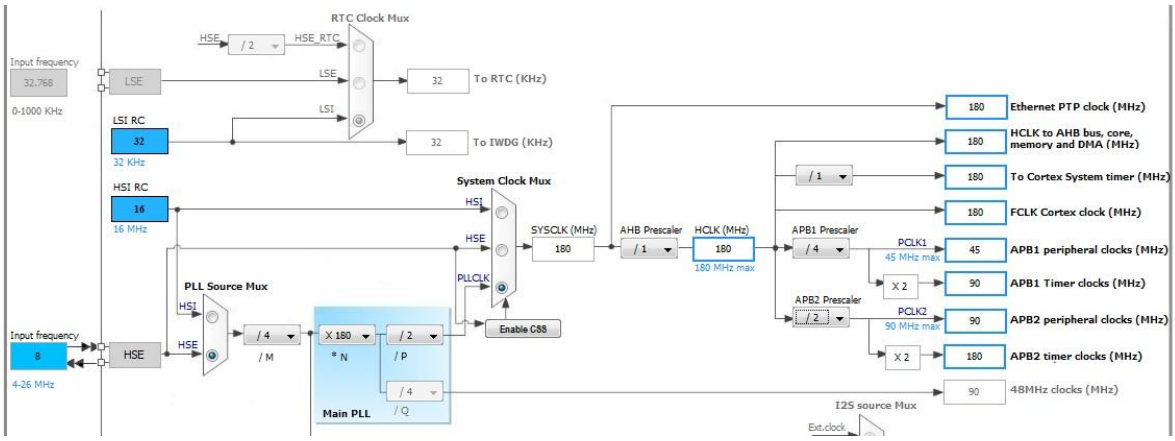


Рисунок 2.9 – Конфігурація системи тактування ARM-МК

Тепер перейдемо до вкладки Configuration для більш розширеного налаштування портів вводу-виводу. В колонці System таблиці обираємо GPIO (General Purpose Input/Output). Виконаємо налаштування кожного піну, що сконфігурований на вивід, тим самим виконаємо перший блок з символом процесу на схемі програми, тобто «Світіння HL0..HL7». Щоб загорівся світлодіод, треба вивести високий логічний рівень на конкретний вивід порту, як показано на рисунку рисунок 2.10.

Як ми можемо переконатися, порти ARM-МК мають дуже гнучкі налаштування:

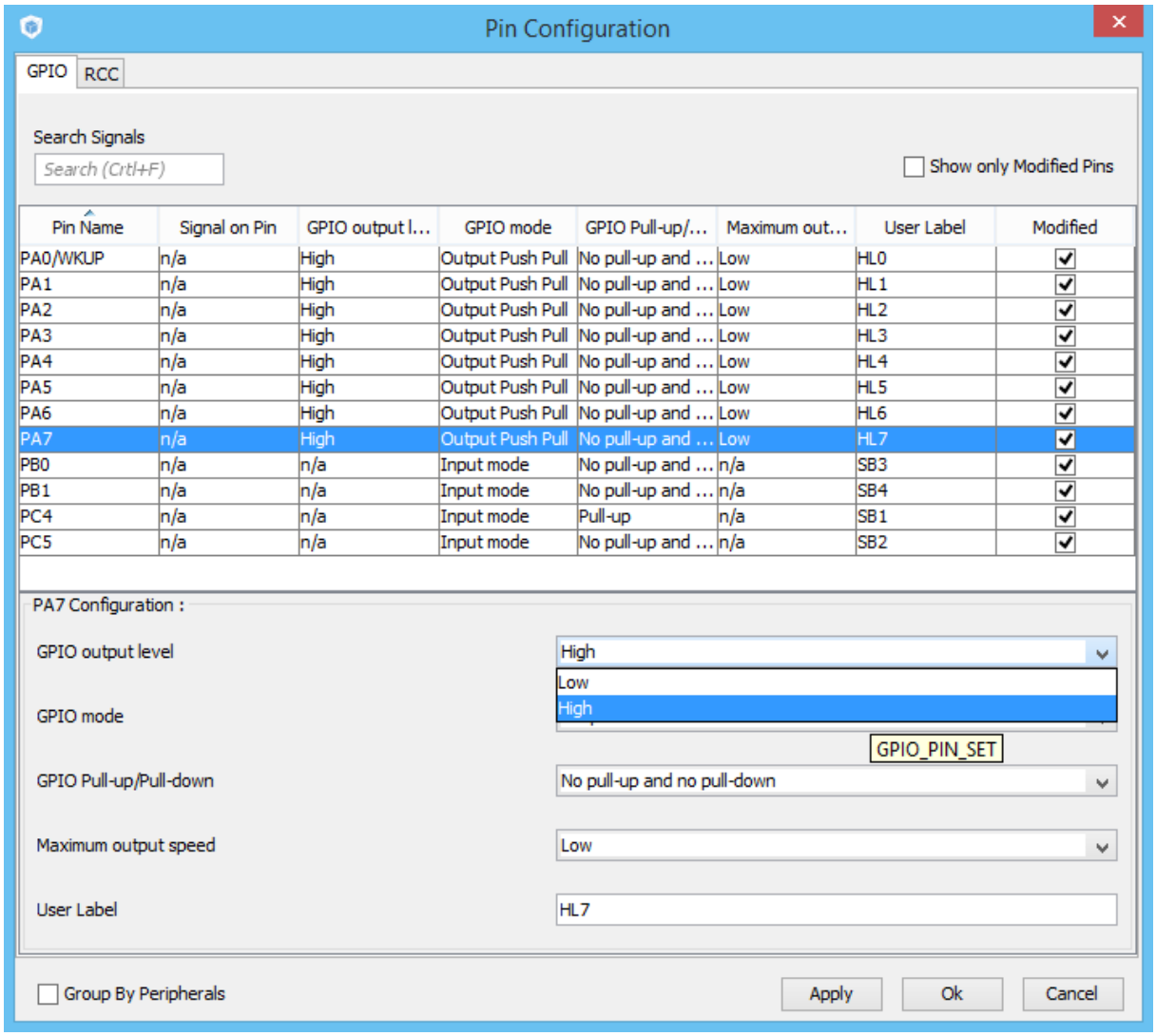


Рисунок 2.10 – Розширені налаштування GPIO

GPIO output level (вихідний рівень) = High/Low (Високий/Низький);

GPIO mode (режим) = Output Push Pull/Output Open Drain (Двотактний / Відкритий стік);

GPIO Pull-up/Pull-down (Підтягуючий до живлення/до землі) = No pull-up and no pull-down/Pull-up/Pull-down;

Maximum output speed (Максимальна швидкість виходу) = Low/Medium/High/Very high (Низька/Середня/Висока/Дуже висока);

User Label () = ?

Останній параметр дозволяє підвищити наочність тексту програми.

Параметр `Maximum output speed` встановлює частоту тактування виводу, а отже, – і швидкість його перемикання. Не будемо забувати, що швидкість та енергоспоживання тісно пов'язані. Зважаючи на вирішення дуже повільних задач інтерфейсу з оператором, залишимо низьку швидкість перемикання виходів. До інших параметрів треба також поставитися уважно, оскільки невідповідний їх вибір може призвести до зайвого енергоспоживання системи.

Оскільки кожна кнопка на стенді підключає лінію порту до землі, потрібний вхід МК підтягнемо до живлення, тобто виберемо параметр Pull-up для ніжки PC4. Це підвищить завадостійкість нашого інтерфейсу.

4. Тепер можна генерувати код ініціалізації: Project->Generate Code (Ctrl+Shift+G). Вводимо шлях для зберігання проекту, називаємо проект і обираємо IDE (у нашому випадку – MDK-ARM V5). Натискаємо Ok.

5. Після генерації проекту можна одразу ж перейти до його відкриття в IDE *Keil MDK-ARM V5*, обравши Open Project.

Таким чином, незважаючи на розгалуженість системи налаштування портів МК, завдяки графічному інтерфейсу *STM32CubeMX* стартовий код формується дуже просто і без необхідності звертання до технічної документації.

Слід зазначити, що *STM32CubeMX* також може автоматично сгенерувати файл звіту проекту (в нашому випадку це буде MPT_Lab2.pdf), який є першим кроком з документування вирішеної задачі. Для цього треба виконати Project->Generate Report (Ctrl+R).

2.4 Підготовка програми в IDE Keil MDK-ARM V5

1. Відкриваємо згенерований проект: Project->Open Project. З'являється дерево проекту, файли користувача у якому знаходяться у групі "Application/User". В інших групах знаходяться файли бібліотеки HAL та драйвери CMSIS. Для перевірки працездатності проекту скомпілюємо його, натиснувши F7. Для вибору апаратного відлагоджувальника *ST-Link* перейдемо до налаштувань проекту (Alt+F7) та на вкладці Debug оберемо Use ST-Link Debugger. За відсутності апаратних засобів на цій вкладці треба вибрати Use Simulator.

2. Для складання програми відповідно до завдання найкраще використати певні функції бібліотеки *HAL* для роботи з портами вводу-виводу, а також функцію програмної затримки. Дані функції можна знайти у відповідних файлах проекту (група Drivers/STM32F4xx_HAL_Driver).

Назва кожного файлу відповідає назві периферії МК, а у самих файлах містяться готові функції користувача для роботи з бібліотекою [16]. Для вирішення нашої задачі знадобляться наступні функції:

```
HAL_GPIO_WritePin(GPIO_TypeDef*  GPIOx,  uint16_t  GPIO_Pin,
GPIO_PinState PinState); // Це функція для виведення логічних
рівнів у порт
HAL_GPIO_ReadPin(GPIO_TypeDef*  GPIOx,  uint16_t  GPIO_Pin);
// Ця функція зчитує логічний рівень на виводі
HAL_Delay(__IO uint32_t Delay);// Функція затримки (у мс), яка
використовує таймер SysTick
```

3. Для вставки власного коду треба акуратно використовувати ті фрагменти вхідних файлів проекту, які помічені заголовками

```
/* USER CODE BEGIN xxx */
…
/* USER CODE END xxx */.
```

Тоді в будь-який момент роботи над проектом можна знову повернутися до програми генерації стартового коду *STM32CubeMX*, виконати додаткові налаштування і повторно згенерувати проект для IDE *Keil MDK-ARM V5*. В цьому випадку всі зміни будуть виконані коректним чином, не торкнувшись нашого коду.

4. У ту частину файлу `main.c`, яка обмежена дужками

```
/* USER CODE BEGIN PV */
/*Private variables */
…
/* USER CODE END PV */
```

додамо визначення змінних:

```
#define HL0_HL7 GPIOA, GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 |
GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7
#define BounceTime 4 // Дзвін контактів, мс
```

5. У іншу частину файлу `main.c` додамо прототип власної функції:

```
/* USER CODE BEGIN PFP */
/* Private function prototypes -------------------------*/
void LED_On_Off(void);
/* USER CODE END PFP */
```

6. З середини відповідних дужок головної функції `main` викликаємо саму функцію:

```
/* USER CODE BEGIN 3 */
     LED_On_Off();
  }
  /* USER CODE END 3 */
```

7. І, нарешті, у ту частину файлу `main.c`, який призначений для розміщення функцій програміста, додамо власний код:

```
/* USER CODE BEGIN 4 */
/** LED_On_Off()
  * @brief  Ця функція перемикає світлодіоди за кнопками.
  * @param  None
  * @retval None
  */ void LED_On_Off(void)
{
// Очікуємо натискання на кнопку SB1:
while(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_4)) {};
          HAL_Delay(BounceTime); // Пригнічуємо дзвін контактів
// Очікуємо відпускання кнопки:
while (!HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_4)) {};
          HAL_Delay(BounceTime); // Пригнічуємо дзвін контактів
// Очікуємо натискання на кнопку SB1:
HAL_GPIO_WritePin(GPIOA,HL0_HL7, GPIO_PIN_RESET);//Гасіння СВД
while(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_4)) {};
          HAL_Delay(BounceTime); // Пригнічуємо дзвін контактів
// Очікуємо відпускання кнопки:
```

```
while (!HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_4)) {};
          HAL_Delay(BounceTime); // Пригнічуємо дзвін контактів
HAL_GPIO_WritePin(GPIOA,HL0_HL7, GPIO_PIN_SET); //Світіння СВД
}    /* USER CODE END 4 */
```

8. Перевіримо працездатність проекту, скомпілюваши його (F7) та завантаживши до стенду Inel-STM безпосередньо з *IDE Keil MDK-ARM V5*: Flash->Dowload (F8).

9. Для відлагодження програми виберемо Debug->Start/Stop Debug Session (Ctrl+F5). Оскільки ми використовуємо безкоштовну версію *MDK-Lite*, на екрані монітора з'явиться попередження (рисунок 2.11). Натиснемо OK і отримаємо можливість з комп'ютера керувати програмою, що міститься в МК.

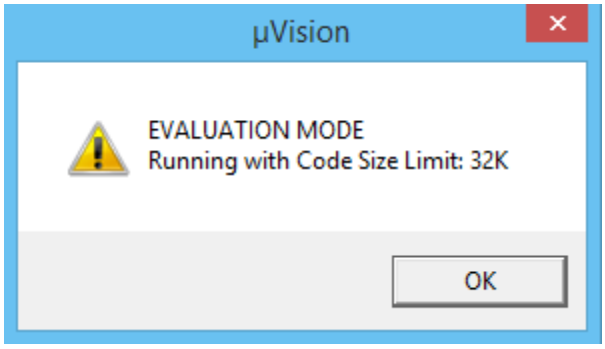


Рисунок 2.11 – Попередження про обмеження за розміром файлу

10. Завантажити власну програму (*hex*-файл) до резидентної пам'яті МК також можна за допомогою утиліти *STM32 ST-LINK Utility*, як це було описано в підрозділі 1.5.

2.5 Відлагодження програми в IDE Keil MDK-ARM V5

1. Оскільки головна функція прикладу розпочинається з виклику функції ініціалізації периферії HAL_Init(), курсор поточного стану виконання програми (подвоєна синьо-жовта стрілка) зупиниться саме на відповідному рядку. Обравши Debug->Step Over (F10), ми завершимо ініціалізацію і перейдемо до наступної функції SystemClock_Config(), яка має налаштувати систему синхронізації МК.

2. Після завершення функції ініціалізації портів введення-виведення MX_GPIO_Init() світлодіоди мають засвітитися, – адже ми раніше в *STM32CubeMX* налаштували потрібні лінії на високий рівень напруги (Рисунок 2.10).

3. Щоб дослідити роботу нашої власної функції LED_On_Off(), доведеться періодично користуватися командою крокування із входом в тіло функції Debug->Step (F11), поєднуючи (для швидкості просування) з вже відомою командою Debug->Step Over (F10). При цьому в деяких точках програми потрібно притримувати кнопку *SB1*.

2.6 Контрольні запитання

1. Наведіть призначення та склад *Keil® MDK*.
2. В яких виданнях доступний *Keil® MDK* та чим вони відрізняються?
3. Для чого призначений, що містить код ініціалізації МК і яким чином можна його створити?
4. Навести порядок встановлення *Keil® MDK*.
5. Які програмні та апаратні ресурси потрібні для відлагодження програми користувача для МК *ARM7* в *Keil® MDK*?
6. Визначте поняття «CMSIS драйвер» і поясніть призначення.
7. Наведіть перелік CMSIS драйверів на сайті виробника *IDE* і прокоментуйте призначення кожного з них.
8. Для чого призначені вкладки у лівому вікні робочого простору `Keil µVision 5`?
9. Наведіть перелік прикладів програм для МК *STM32F429ZGTx* на сайті виробника *IDE* і прокоментуйте призначення кожного з них.
10. Які налаштування *Keil® MDK* потрібно зробити для спільної роботи з стендом *Inel-STM*?
11. Визначте поняття «*HAL драйвер*» і поясніть призначення.
12. Поясніть аргументи *HAL* драйверів `HAL_GPIO_WritePin()`, `HAL_GPIO_ReadPin()` та `HAL_Delay()`.
13. До яких фрагментів вхідних файлів автоматично згенерованого *STM32CubeMX* проекту можна вставляти власний код?
14. З чим пов’язана відмінність тексту від схеми програми прикладу?
15. Яку функцію *HAL* треба використати у другому блоці схеми програми (Рисунок 2.8), щоб при вирішення тієї ж задачі можна було відмовитися від останніх п’яти блоків?

2.7 Хід роботи

2.7.1 Підготовчі стадії

Поза навчальною лабораторією треба зробити наступне:

1. Виконайте дії, описані в підрозділах 2.2 та 2.3.
2. Оформити першу частину звіту з виконання лабораторної роботи, в якій зазначити прізвища студентів бригади та дати письмові відповіді на контрольні запитання (п. 2.6).
3. Покиньте MDK та запустіть генератор коду ініціалізації *STM32Cube*.

2.7.2 Етап розробки програмного забезпечення

1. Виконайте дії, описані в підрозділі 2.4. Новий проект з назвою `MPT_Lab2` в генераторі коду ініціалізації *STM32Cube* треба створити саме для роботи в *IDE µVision*. Для цього на вкладці `Project->Settings STM32CubeMX` у вікні `Toolchain/IDE` треба задати `MDK-ARM V5`.

2. Після вибору `Project->Generate Code` та успішної генерації стартового коду на запит про те, що ж робити, треба обрати `Open Project`. Це має привести до запуску *IDE µVision*.

3. Відповідно до отриманого варіанту завдання (підрозділ 2.9) розробіть схему, вхідний текст та відлагодьте програму для Inel-STM. Під час перевірки дотримуйтесь порядку дій, як у підрозділі 2.5.

2.8 Орієнтовні варіанти завдань

1. Після запуску програми мають засвітитися усі одиничні індикатори HL7…HL0. За натисканням кнопки SB1 починає відтворюватися вогник, що біжить у напрямку HL7 -> HL0 із затримкою на кожен крок світіння індикатора у 100 мс. За повторним натисканням кнопки SB1 знову мають світитися усі індикатори.

2. Після завантаження програми до МК усі одиничні індикатори HL7…HL0 не світяться. За натисканням кнопки SB2 починає відтворюватися вогник із двох сусідніх світлодіодах, що біжить у напрямку HL0 -> HL7 із затримкою на кожен крок світіння індикатора у 190 мс. За повторним натисканням кнопки SB2 знову мають згаснути усі індикатори.

3. Після завантаження програми до МК усі одиничні індикатори HL7…HL0 не світяться. За натисканням кнопки SB3 починає відтворюватися тінь, що біжить у напрямку HL0 -> HL7 із затримкою на кожен крок світіння індикатора у 150 мс. За повторним натисканням на кнопку SB3 напрям руху тіні має змінюватися.

4. Після завантаження програми до МК одиничні індикатори HL7…HL0 світяться через один. За натисканням кнопки SB4 починає відтворюватися рух цих вогників у напрямку HL7 -> HL0 із затримкою на кожен крок світіння індикатора у 180 мс. За повторним натисканням кнопки SB2 знову усі одиничні індикатори HL7…HL0 горять через один.

5. Після завантаження програми до МК починає відтворюватися два бігучих вогники: HL7 -> HL4 і HL0 -> HL3 із затримкою на кожен крок світіння індикатора у 90 мс. За натисканням кнопки SB1 міняється напрям вогників: HL4 -> HL7 і HL3 -> HL0. За повторним натисканням кнопки SB1 виконується початкова анімація.

6. Після завантаження програми до МК усі одиничні індикатори HL7…HL0 не світяться. За натисканням кнопки SB2 на світлодіодах починає відтворюватися додатний двійковий лічильник із затримкою на кожен крок світіння індикатора у 50 мс. За повторним натисканням кнопки SB2 знову мають згаснути усі індикатори.

7. Після завантаження програми до МК усі одиничні індикатори HL7…HL0 не світяться. За натисканням кнопки SB3 на світлодіодах починає відтворюватися стовпчик, що засвічується у напрямку від HL0 до HL7 із затримкою на кожен крок світіння індикатора у 120 мс. За повторним натисканням кнопки SB3 знову мають згаснути усі індикатори.

8. Після завантаження програми до МК усі одиничні індикатори

HL7…HL0 не світяться. За кожним натисканням кнопки SB4 відтворюється додатний лічильник, а за умови утримання цієї кнопки більше 1 с зміна стану має відбуватися автоматично з кроком 200 мс. За натисканням кнопки SB1 лічильник має обнулитися.

9. Після завантаження програми до МК усі одиничні індикатори HL7…HL0 не світяться. За кожним натисканням кнопки SB2 відтворюється лічильник Грея, а за умови утримання цієї кнопки більше 0,5 с зміна стану має відбуватися автоматично з кроком 250 мс. За натисканням кнопки SB3 лічильник має обнулитися.

10. Після завантаження програми до МК усі одиничні індикатори HL7…HL0 світяться. За натисканням кнопки SB2 на світлодіодах починає відтворюватися стовпчик, що загасає у напрямку від HL7 до HL0 із затримкою на кожен крок світіння індикатора у 230 мс. За повторним натисканням кнопки SB2 знову мають світитися усі індикатори.

2.9 Вимоги до звіту по роботі

Звіт про виконання роботи повинний містити:

1. Титульний аркуш встановленого в університеті зразку.

2. Письмові відповіді на контрольні запитання.

3. Схему програми вирішеної задачі з функціональною схемою підключення застосованих індикаторів та кнопок.

4. Текст програми з докладними коментарями, які відповідають схемі програми.

5. Висновки по роботі, в яких зазначити, чи досягнута мета роботи. Що саме конкретно (перерахувати по пунктах) дозволяє досліджувати *Keil® MDK* спільно з лабораторним стендом Inel-STM?

3 Лабораторна робота №3. Дослідження таймерів, системи переривань та широтно-імпульсних модуляторів

Мета роботи: практично дослідити можливості резидентних таймерів, принципи організації переривань та особливості генераторів ШІМ-сигналу у ARM-МК *STM32F429ZGT6*.

3.1 Особливості резидентних таймерів в ARM-МК

У контролерів серій STM32F4 таймери поділяють на декілька груп.

3.1.1 Базові таймери (*Basic Timers*)

Це лічильники, схожі на системний таймер *SysTick*, проте більш гнучкі. Вони не призначені для керування виводами (пінами) МК. У МК *STM32F429ZGT6* до базових таймерів належать 16-розрядні *TIM6* та *TIM7*. Структура базового таймера показана на рисунку 3.1.

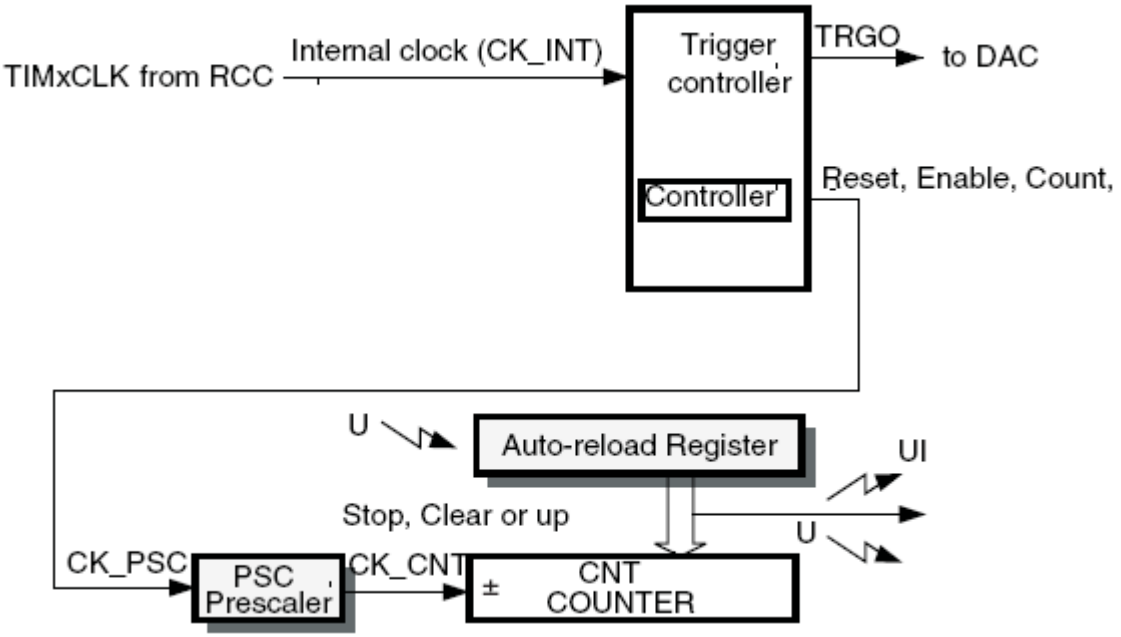


Рисунок 3.1 – Структура базового таймера

Основне призначення базових таймерів *TIM6* та *TIM7* – запуск цифро-аналогових перетворювачів (ЦАП) за допомогою виходів *TRGO* кожного з таймерів. Однак базові таймери можна використовувати і для відліку часу, економлячи більш просунуті таймери.

Таймери мають лічильні регістри *TIMx_CNT* (де *x* – номер таймера), які за кожного такту зростають на 1. Внутрішні тактові імпульси поступають на лічильник через 16-розрядний регістр *TIMx_PSC*, до якого треба записати число в діапазоні від *0* (коефіцієнт ділення *1*) до *65535* (*65536*).

Коли лічильник досягає значення, збереженого в регістрі *TIMx_ARR* (*Auto-Reload Register*), відбувається одразу декілька речей: до *TIMx_CNT* завантажується 0, генерується подія переповнення лічильника (*counter overflow*) та подія оновлення (*update event*). Це дві різні події, які в просунутих таймерах проявляються більш виразно. Проте і для базових таймерів

подію оновлення можна згенерувати незалежно (наприклад, встановивши біт *UG* регістра *TIMx_EGR* (*event generation register*).

У відповідь на ту чи іншу подію може генеруватися переривання або запит на прямий доступ до пам'яті (DMA).

В регістри можна писати і під час роботи, однак, за замовчуванням, використовуються "тіньові" копії (*shadow registers*), а оновлення основних регістрів відбувається після подій.

CMSIS надає зручні імена для регістрів та бітів. Наприклад, щоб звернутися до лічильного регістра таймера *TIM7*, треба написати:

`TIM7->CNT;`

бітова маска (тобто номер біта в регістрі) для біта *UG* має назву `TIM_EGR_UG`.

Для встановлення цього біту можна використати наступний оператор `TIM7->EGR |= TIM_EGR_UG`.

Інші регістри для роботи з базовими таймерами:

1. *TIMx_CR1* – головний регістр керування. Важливі біти:

CEN (*Counter ENable*). Запуск відліку;

UDIS (*Update DISable*). Заборона події оновлення;

URS (*Update Request Source*). Дозволяє заборону оновлення за зміною біту *UG*;

OPM (*One-Pulse Mode*). Якщо 1, після події оновлення таймер зупиняється, очищаючи біт *CEN*;

ARPE (*Auto-Reload Preload Enable*). Керує буферізацією *ARR* – чи він ступає в дію одразу (0) чи після події оновлення (1).

2. *TIMx_CR2* – другий регістр керування. За допомогою бітів *MMS* (*Master Mode Selection*) дозволяє обирати, яким саме чином цей таймер керує підлеглим. Важливим значенням є 010b (*Update*): підлеглий таймер "смикається" за подією оновлення. При *MMS* = 000b (*Reset*) за встановленням *UG* підлеглий таймер оновлюється.

3. *TIMx_DIER* (*DMA/Interrupt Enable Register*). Керування генерацією переривань та запитів *DMA*. Біти *UDE* (*Update DMA request Enable*) та *UIE* (*Update Interrupt Enable*), встановлені у 1, дозволяють відповідну реакцію.

4. *TIMx_SR* – регістр статусу. Біт *UIF* вказує, чи була подія оновлення.

3.1.2 Таймери загального використання (*General purpose*)

Ці таймери крім відліку дозволяють порівнювати вміст лічильного регістра з наперед заданим кодом і, у момент зрівняння двох цифрових значень, створювати подію. Такою подією може бути, наприклад, зміна стану виводу (ніжки, піну) МК. Такий режим роботи таймера в документації називається *Output Compare* і використовується для створення затримок, формування імпульсів, генерації періодичних сигналів з широтно-імпульсною модуляцією (*ШІМ, Power Width Modulation – PWM*). Особливий випадок становить режим захоплення за входом (*Input Capture*), коли

подія на ніжці МК (наприклад, перехід з нуля до одиниці) призводить до запису поточного стану лічильного регістра до спеціального регістра захоплення. Цей режим можна використати для визначення тривалості імпульсу, частоти вхідного сигналу, синхронізації з периферією – енкодерами, датчиками Холла тощо. Для підтримки режимів порівняння та захоплення в таймерах загального використання передбачають декілька спеціальних регістрів *TIMx_CCRy* (*TIMx capture/compare register y*).

Кожен таймер загального використання має один лічильник, проте підтримує багато каналів, якими може керувати одночасно. Різні такі таймери мають різний набір можливостей із цієї множини.

У МК *STM32F429ZGT6* до таймерів загального використання належать 16-розрядні *TIM3*, *TIM4* та 32-розрядні *TIM2*, *TIM5*.

Особливості таймерів загального використання:

– підтримують 4 канали;
– не мають комплементарних виводів;
– не можуть працювати з повторами (*repetitions*), відповідно, не мають регістра *TIMx_RCR*;
– не підтримують режим та не мають входів переривання.

3.1.3 Просунуті таймери (*Advanced Timers*)

По-перше, кожен з просунутих таймерів має весь набір можливостей таймерів загального використання. По-друге, вони мають додаткові можливості, – такі, як засоби керування двигунами, джерелами живлення, "аварійною зупинкою" (*break*) тощо.

У МК *STM32F429ZGT6* до просунутих таймерів належать 16-розрядні *TIM1* та *TIM8*.

Особливості просунутих таймерів:

– лічильник може рахувати вгору, вниз, а також туди, потім назад;
– як і в базових таймерах, присутній подільник частоти (*prescaler*);
– мають до 6 каналів, на кожному з яких можна незалежно захоплювати сигнал, здійснювати порівняння і виведення, генерувати ШІМ, працювати в одно-імпульсному режимі;
– комплементарні виводи із затримкою між ними (*dead-time*) – виводи, сигнал на яких має протилежне значення, причому можна програмно встановити затримку переключення у новий стан – *dead time*;
– як і в базових таймерах є можливість тактувати один таймер іншим, об'єднуючи їх у ланцюжки;
– лічильник повторів, який дозволяє реагувати після вказаної кількості повторів події (переповнення, наприклад);
– один-два так званих *break inputs* (аварійні входи, по сигналу на яких виводи таймерів набувають вказаних безпечних значень);
– засоби підтримки енкодерів та датчиків Холла;
– також можуть активувати АЦП чи ЦАП у відповідь на події.

3.1.4 Канальні таймери (*Channel Timers*)

Додатково виділяють канальні таймери (*Channel Timers*) та канальні таймери з комплиментарними виходами (*Channel Timers with Complementary Output*), які мають той же набір властивостей, що і таймери загального використання, але керують меншою кількістю каналів.

У МК *STM32F429ZGT6* є чотири 16-розрядні одноканальні таймері *TIM10*, *TIM11*, *TIM13*, *TIM14* та два двоканальні *TIM9*, *TIM12*.

Важливо те, що побудова усіх таймерів спільна: більш потужні є розширеними варіантами примітивніших [4, 7, 12, 17].

3.2 Генерація сигналів з широтно-імпульсною модуляцією в ARM-МК

Генерація ШІМ, фактично, є одним з способів використання режиму *Output compare* в таймерах. Вміст регістру *TIMx_ARR* задає період, *TIMx_CCRx* – тривалість заповнення періоду високим рівнем. При полярності із активним високим заповнення визначається, як *TIMx_ARR* – *TIMx_CCRx*). Якщо відлік відбувається вгору чи вниз, ШІМ вирівняний по краю, якщо туди, потім назад, – по центру.

Приклад генерації ШІМ-сигналу, який вирівняний по краю при рахунку вгору, показаний на рисунку 3.2. Таймер керує виводом МК *OCXREF*. *CCxIF* – це біт, який вказує, що відбулася подія порівняння. При встановленні цього біту в 1 та за умови відповідного дозволу, генерується переривання чи запит на *DMA*. У прикладі *TIMx_ARR* = 8.

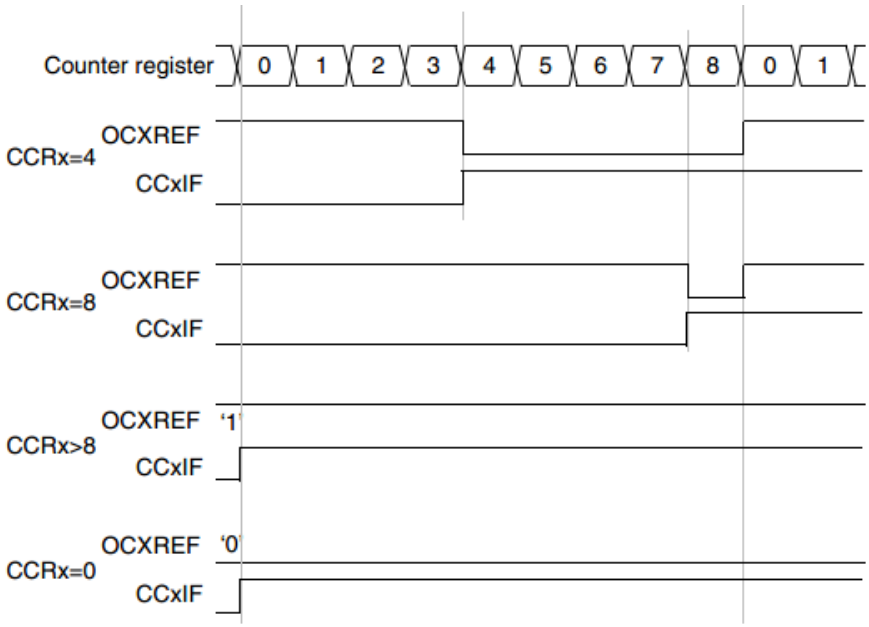


Рисунок 3.2 – Діаграми генерації апаратного ШІМ-сигналу

Більш докладно з можливостями, структурою, режимами роботи, регістрами та настрояками таймерів у складі МК *STM32F429ZGT6* можна ознайомитися в [7].

3.3 Приклад використання таймерів, ШІМ та переривань

Розглянемо процедуру вирішення наступної задачі. Треба плавно (протягом 1 *с*) періодично змінювати яскравість одного з чотирьох світлодіодів від нуля до максимально можливого значення, а потім, так само, – від максимуму до нуля. За допомогою натискання на кнопку *SB*1 має відбуватися почерговий перехід від попереднього світлодіода до наступного.

Для вирішення задачі будемо використовувати 4 канали ШІМ на частоті сигналу 1 *кГц*, змінюючи в них шпаруватість від 0 до 100 протягом однієї секунди, а потім, за цей самий час, зменшуючи від 100 до 0. Натискання на кнопку має змінювати канал ШІМ (вимикати попередній світлодіод і вмикати наступний). Для реакції на кнопку використаємо зовнішнє переривання. Спробуємо розмістити код програми у обробниках переривань від таймера та зовнішнього переривання.

На рисунках 3.3 – 3.5 наведені схеми програмних блоків. Символом “Q” позначений коефіцієнт заповнення періоду ШІМ-сигналу.

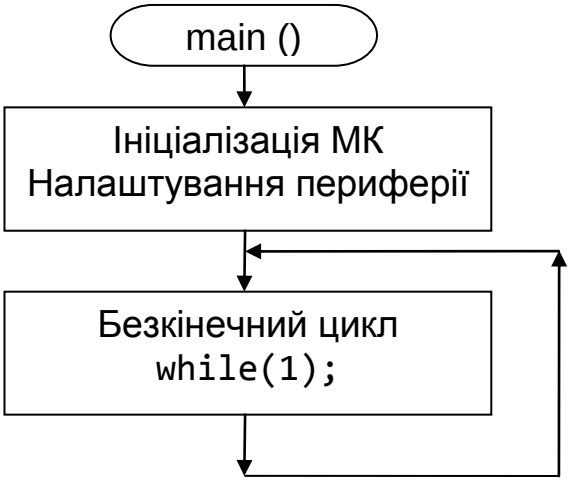


Рисунок 3.3 – Схема функції `main ()`

Рисунок 3.4 – Схема функції обробника зовнішнього переривання (SB1)

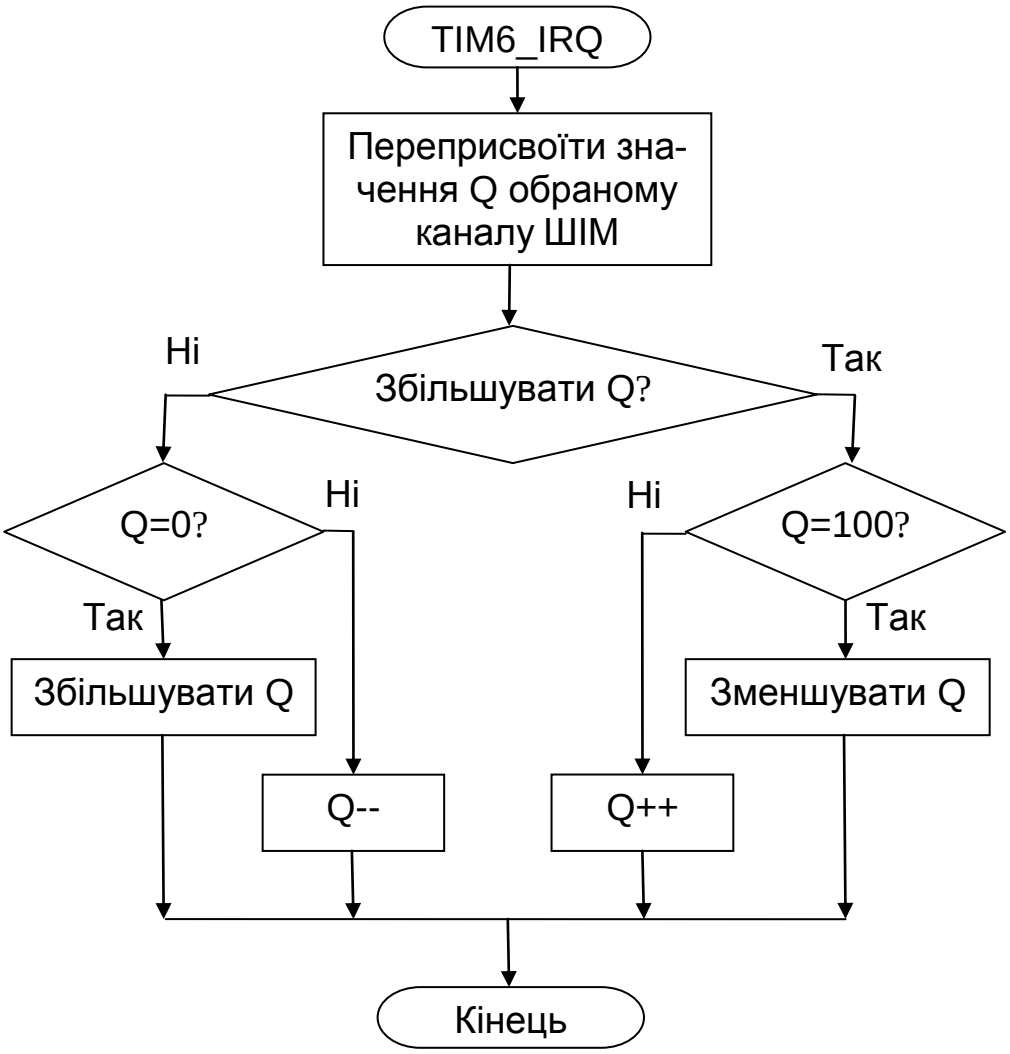


Рисунок 3.5 – Схема функції обробника переривання від Таймера 6

3.3.1 Створення коду ініціалізації для МК STM32F429ZGT6

1. Створимо новий проект у програмі *STM32CubeMX* з назвою, наприклад, *MPT_Lab3* і активуємо тактування від кварцового резонатора, обравши в групі RCC:

`High Speed Clock (HSE) = Crystal/Ceramic Resonator.`

2. Налаштування таймерів розпочнемо з каналів ШІМ Таймера 5 (група `TIM5`), які керують виводами МК, і до яких підключено світлодіоди HL0…HL3 (рисунок 3.6).

3. Налаштуємо роботу із кнопкою *SB*1, призначивши лінії порту *PC*4 зовнішнє переривання, як показано на рисунку 3.6.

4. Проведемо налаштування системи синхронізації МК на вкладці `Clock Configuration` аналогічно тому, як ми це робили в підрозділі 2.3.

5. Також активуємо *Таймер* 6 (`TIM6`, рисунок 3.6) для виконання команд із зміни яскравості світлодіодів. Власне налаштування таймера про-

ведемо на вкладці Configuration->Control->TIM6, як показано на рисунку 3.7. При цьому враховуємо, що таймер тактується сигналом APB1 Timer clocks частотою 90 *МГц*. Нам потрібна частота спрацювання переривання у 100 *Гц*.

6. У вкладці NVIC Settings вікна TIM6 Configuration увімкнемо переривання від даного таймера (Enabled).

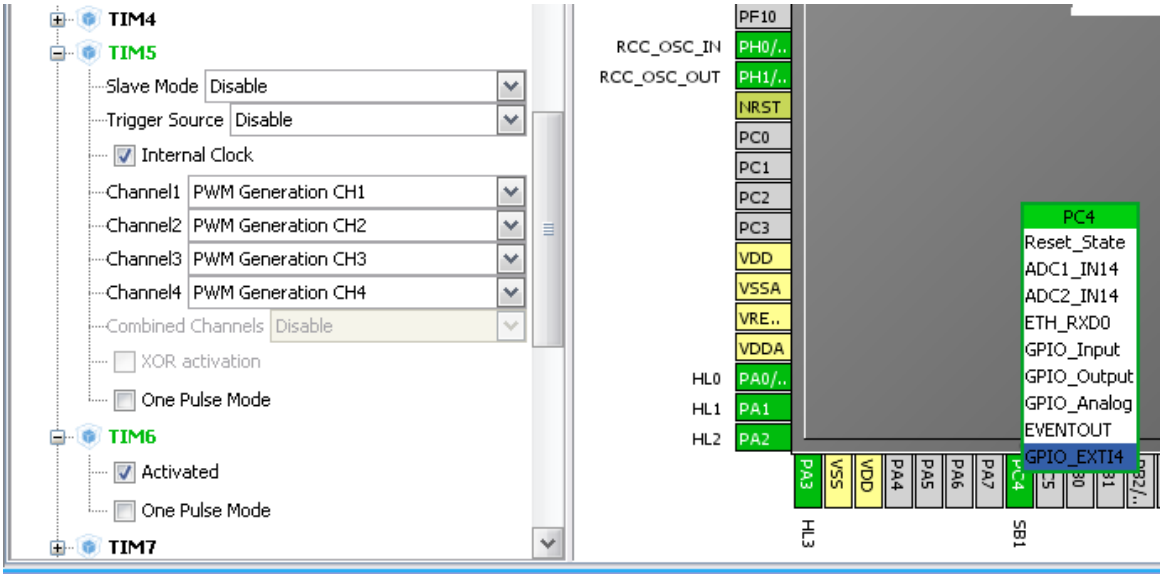


Рисунок 3.6 – Налаштування таймерів та переривання EXTI4

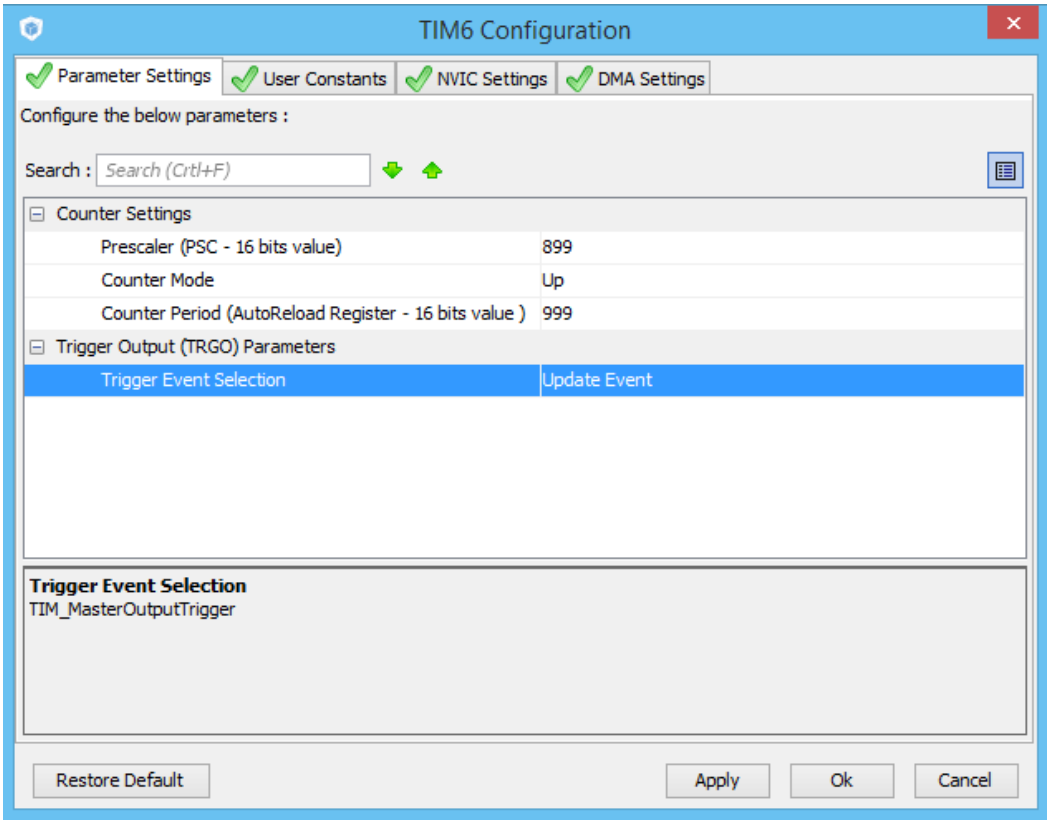


Рисунок 3.7 – Налаштування TIM6

7. Виконаємо налаштування таймера 5 для генерації ШІМ-сигналу,

як показано на рисунку 3.8.

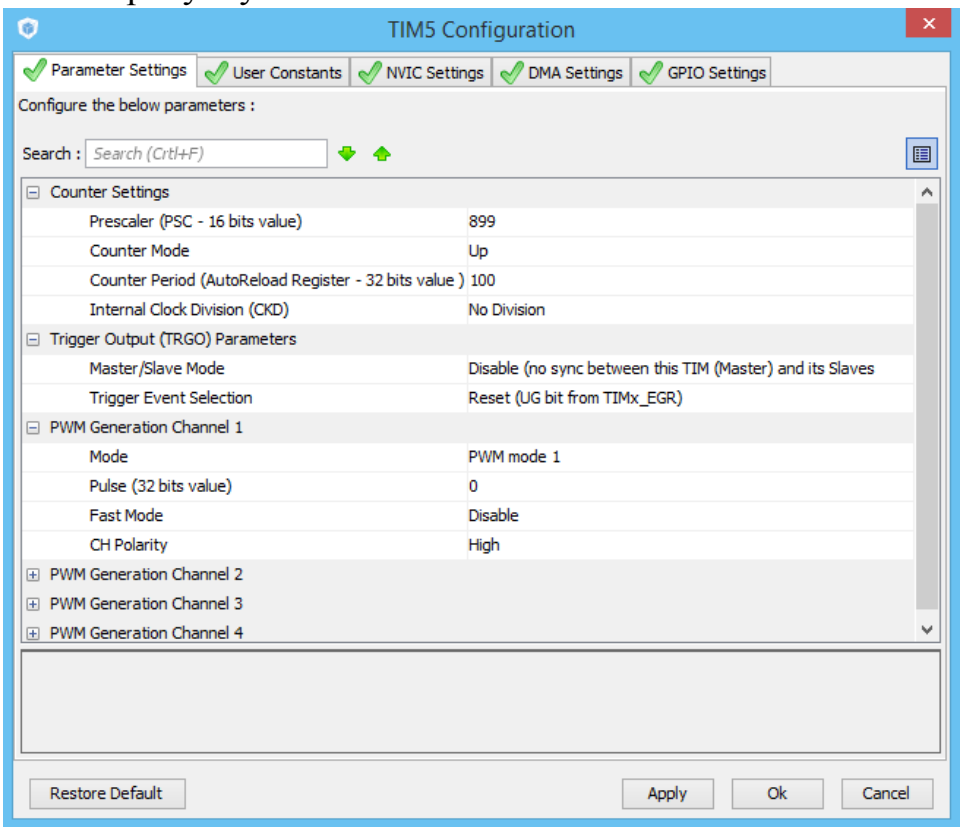


Рисунок 3.8 – Налаштування TIM5

8. Порт PC4 аналогічно п. 4 підрозділу 2.3 підтягнемо до живлення (Pull-up) та залишимо саме запропонований варіант реагування при відпусканні кнопки (GPIO mode->External Interrupt Mode with Rising edge trigger detection).

9. Оскільки лінії PA0…PA3 тепер виконують альтернативні функції (обслуговують Таймер 5), їхнє налаштування можна виконати, знову ж таки, у вікні TIM5 Configuration на вкладці GPIO Settings. Тут мабуть що можна проставити мітки ніжок HL0…HL1 для підвищення наочності тексту програми.

10. Наостанок зайдемо у вікно налаштування вкладених векторизованих переривань:

Configuration -> NVIC Configuration

На вкладці NVIC переконаємося у наявності дозволів на обслуговування потрібних переривань:

EXTI line4; TIM5 Global interrupt -> Yes.; TIM6 Global interrupt… -> Yes.

На вкладці Code generation дамо вказівку на генерацію заготовок тільки тих обробників переривань, які потрібні в нашому проекті (рисунок 3.9). Це зменшить обсяг файлу з обробниками. До речі, на цій же вкладці легко керувати порядком розміщення обробників, що залишаться.

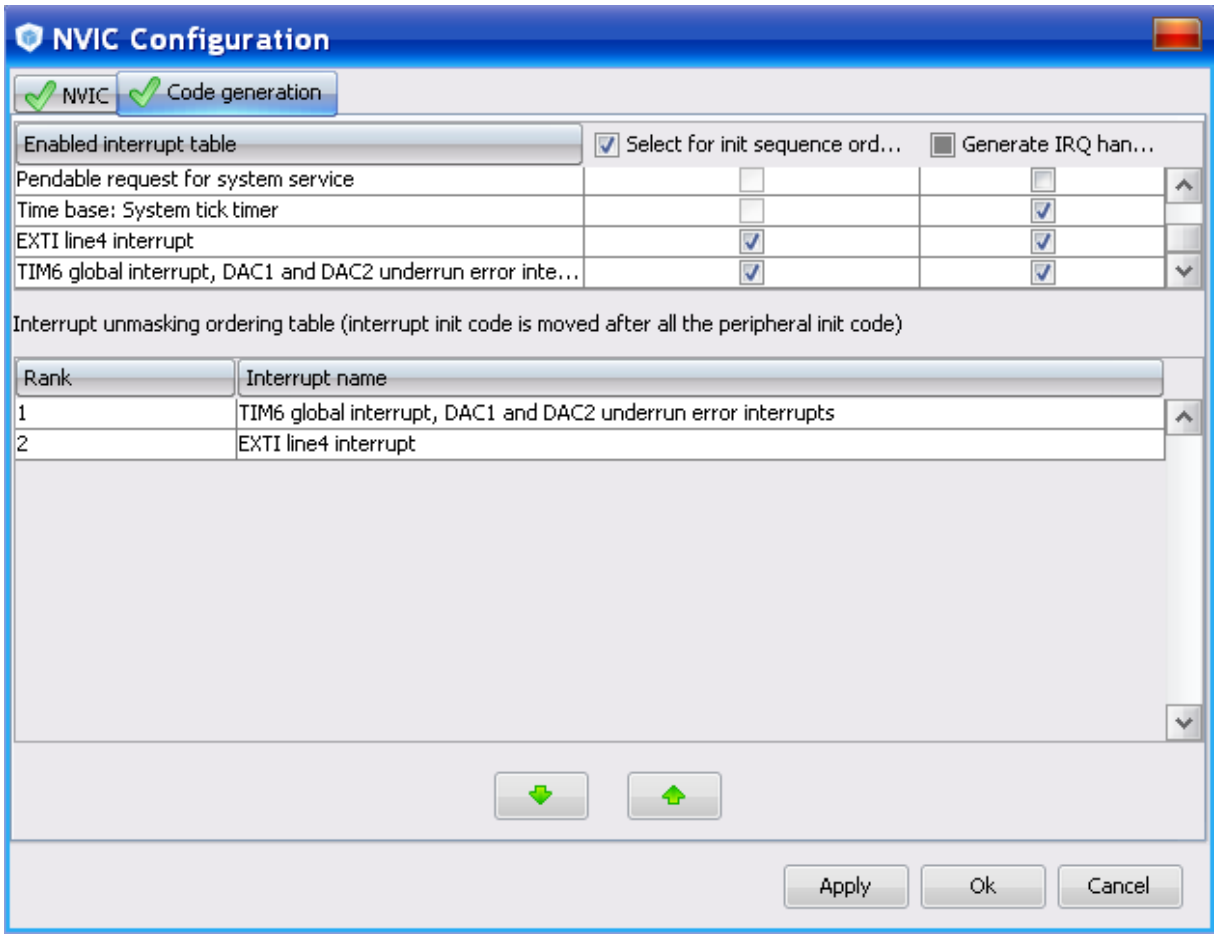


Рисунок 3.9 – Налаштування генерації коду обробників переривань

Примітка. У нашому прикладі переривання від `TIM5` не використовуються. На вкладці `NVIC` дозвіл дано з метою реалізації можливості керування з середини одного переривання (`TIM6`) іншим ресурсом (`TIM5`). В даному випадку опція всього лише дозволяє автоматично генерувати декларацію змінної `htim5`, яка є зовнішньою для файлу, в якому розміщуються обробники переривань. Тобто для того, щоб *CubeMX* зміг генерувати рядок (який також можна написати вручну):

```
extern TIM_HandleTypeDef htim5;
```

11. Генеруємо код ініціалізації: `Project->Generate Code (Ctrl+Shift+G)` та відкриваємо його в IDE *Keil MDK-ARM V5*, обравши `Open Project`.

Аби зменшити обсяг повторюваних від проекту до проекту файлів, в налаштуваннях *STM32CubeMX* можна, наприклад обрати:
`Project->Settings (Alt+P) -> Code Generator -> Copy only the necessary library files`.

Більш радикальний варіант – це обрати
`Add necessary library files as reference in the toolchain project configuration file`.

3.3.2 Підготовка програми в IDE Keil MDK-ARM V5

1. З середини відповідних дужок головної функції `main` (після ініціалізації периферії) однократно викликаємо дві функції *HAL*:

```
  /* USER CODE BEGIN 2 */
// Вмикаємо таймер 6 і переривання від нього
HAL_TIM_Base_Start_IT(&htim6);
// Вмикаємо генерацію ШІМ на першому каналі Таймера 5
HAL_TIM_PWM_Start(&htim5, TIM_CHANNEL_1);
  /* USER CODE END 2 */
```

Нескінчений цикл головної функції `main` залишаємо без змін.

2. Відкриваємо файл `stm32f4xx_it.c`, який містить обробники переривань нашого проекту (`Interrupt Service Routines`). Всередині відповідних дужок вводимо код:

```
/* USER CODE BEGIN 0 */
//Змінна для зберігання коефіцієнта заповнення
volatile uint8_t DutyCycle = 0;
enum LEDs
{
     LED_0,
     LED_1,
     LED_2,
     LED_3
};
enum Dir
{
     UP,
     DOWN
};
// Змінна для вибору каналу ШІМ
enum LEDs LED = LED_0; // Розпочинаємо з першого світлодіода
// Змінна для вибору напрямку зміни коефіцієнта заповнення
enum Dir Drc = UP; // Розпочинати будемо з 0 вгору
/* USER CODE END 0 */
// -------------------------------------------------------------------------------------
  /* USER CODE BEGIN EXTI4_IRQn 1 */
// Що робити за натисканням кнопки
     switch(LED)
     {
          case LED_0:
//Вимикаємо попередній канал ШІМ
               HAL_TIM_PWM_Stop(&htim5, TIM_CHANNEL_1);
               LED = LED_1; //змінюємо канал
               HAL_TIM_PWM_Start(&htim5, TIM_CHANNEL_2);
//Вмикаємо наступний канал ШІМ
          break;
```

```
          case LED_1:
               HAL_TIM_PWM_Stop(&htim5, TIM_CHANNEL_2);
               LED = LED_2;
               HAL_TIM_PWM_Start(&htim5, TIM_CHANNEL_3);
          break;

          case LED_2:
               HAL_TIM_PWM_Stop(&htim5, TIM_CHANNEL_3);
               LED = LED_3;
               HAL_TIM_PWM_Start(&htim5, TIM_CHANNEL_4);
          break;

          case LED_3:
               HAL_TIM_PWM_Stop(&htim5, TIM_CHANNEL_4);
               LED = LED_0;
               HAL_TIM_PWM_Start(&htim5, TIM_CHANNEL_1);
          break;

          default: break;
     }
     DutyCycle = 0; // Обнулити значення коефіцієнта заповнення
  /* USER CODE END EXTI4_IRQn 1 */
// -------------------------------------------------------
  /* USER CODE BEGIN TIM6_DAC_IRQn 1 */
// Що робити за перериванням від Таймера 6
     switch(LED)    // Залежить від номера світлодіода
     {
          case LED_0:
//Записати коефіцієнт заповнення до регістру порівняння
               TIM5->CCR1 = DutyCycle;
          break;
          case LED_1:
               TIM5->CCR2 = DutyCycle;
          break;
          case LED_2:
               TIM5->CCR3 = DutyCycle;
          break;
          case LED_3:
               TIM5->CCR4 = DutyCycle;
          break;
          default: break;
     }
     Switch (Drc)   // В залежності від напряму зміни
     {
     case UP: //Якщо зміна коефіцієнта заповнення вгору
```

```
                if(DutyCycle == 100) // Досягли максимуму?
//Змінити напрям зміни коефіцієнту заповнення на протилежний
                {    Drc = DOWN;    }
     else //Інкремент коефіцієнта заповнення ШІМ-сигналу
                {DutyCycle++;  }
          break;
     case DOWN: //Якщо зміна коефіцієнта заповнення вниз
                if(DutyCycle == 0) {Drc = UP;}//Мінімум?
     else {DutyCycle--;}//Декремент коефіцієнта заповнення
          break;
     default: break;
     }
  /* USER CODE END TIM6_DAC_IRQn 1 */
```

3.4 Контрольні запитання

1. За якими типами розрізняють таймери в МК *STM32F4*? В чому полягають їхні ключові особливості?

2. Для чого потрібна та яким чином реалізується широтно-імпульсна модуляція в МК *STM32F4*?

3. Як перевести МК в режим зниженого енергоспоживання?

4. Схарактеризуйте режими роботи таймерів в прикладі підрозділу 3.3. В яких рядках програми відбувається встановлення цих режимів? Як запустити таймер? Як зупинити таймер?

5. Як за допомогою таймера з використанням та без переривання згенерувати затримку?

6. Як за допомогою таймера можна виміряти тривалість імпульсу?

7. Яку частоту подає на вхід вузла лічильника `TIM5` система синхронізації МК у розглянутому прикладі? Чому дорівнює період ШІМ-сигналів T_{PWM}, які формує лічильник `TIM5`? Яка дискретність зміни ширини імпульсів Δt_{PWM}? В яких рядках програми відбувається встановлення зазначених параметрів?

8. В якому режимі та на якій частоті працює лічильник `TIM6`?

3.5 Хід роботи

1. Вивчіть особливості таймерів та системи переривань *STM32F429ZGT6*: номенклатура, узагальнена структура, функціональні можливості та настроювання [7, 8, 12, 17].

2. Оформіть першу частину звіту з виконання лабораторної роботи, в якій дайте письмові відповіді на контрольні запитання.

3. Виконайте дії, описані в підрозділі 3.3.

4. Безпосередньо у викладача отримайте допуск до виконання роботи в лабораторії та варіант завдання.

5. Розробіть алгоритм, складіть програму (згідно з варіантом), перевірте її роботу на стенді і продемонструйте викладачеві.

3.6 Орієнтовні варіанти завдань

1. Варіанти завдань повністю відповідають підрозділу 2.9, але але часові затримки реалізовані за допомогою переривань від таймерів, а обробка натискань на кнопки здійснюється за допомогою переривань. Усі ітерації анімацій із одиничними індикаторами виконуються у перериванні від таймеру *TIM6*, який налаштований на необхідну частоту згідно завдань.

2. Підключити будь-який світлодіод до ШІМ-каналу таймеру *TIM4*. Регулювати яскравістю світіння за допомогою кнопки.

3.7 Вимоги до звіту по роботі

Звіт повинен містити функціональну схему підключення застосованих апаратних засобів до МК з вказівкою конкретних виводів, завдання, алгоритми і коментовані тексти розроблених програм. Необхідно також навести опис функцій та їхніх параметрів, блоків даних, зробити висновки щодо досягнутої мети роботи.

4 Лабораторна робота №4. Дослідження методів введення аналогової інформації та зв'язку з персональним комп'ютером в МПС на базі STM32F4

Мета роботи: практично дослідити аналогово-цифрове перетворення та зв'язок з комп'ютером через UART в мікропроцесорній системі на базі ARM-МК *STM32F429ZGT6*.

4.1 Особливості резидентних АЦП в МК STM32F4

До МК *STM32F429ZGT6* вбудовані три 12-розрядні незалежні потужні модулі аналого-цифрового перетворення (АЦП), кожен з яких підтримує до 16 зовнішніх каналів. Для усіх каналів можна виконувати або однократні перетворення, або працювати в режимі сканування, коли для обраної групи аналогових входів виконується автоматичне перетворення. Необхідність в цьому модулі виникає достатньо часто, оскільки існує багато джерел та датчиків саме аналогових сигналів.

Додаткові логічні функції, вбудовані в інтерфейс АЦП, дозволяють виконувати:

синхронну вибірку на декількох каналах і зберігання даних;

почергові вибірки каналів і зберігання отриманих даних.

Внутрішню будову модуля АЦП показано на рисунку 4.1.

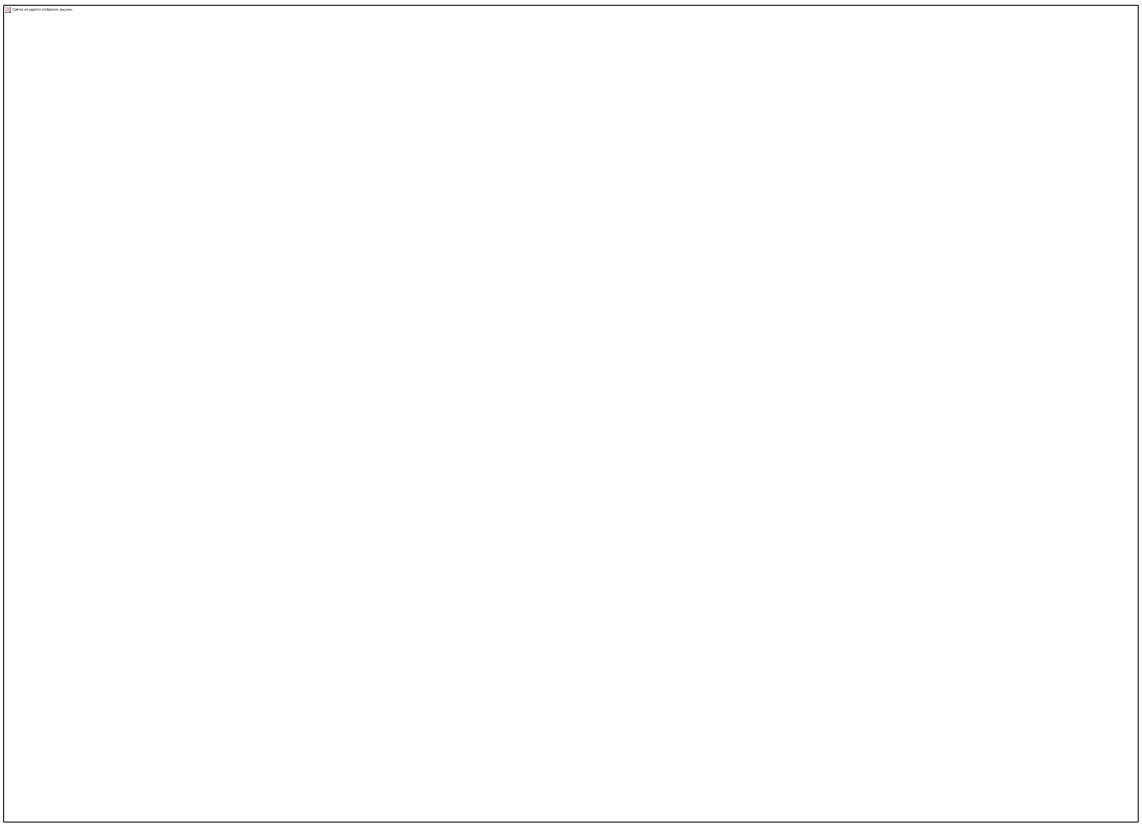


Рисунок 4.1 – Функціональна схема модуля АЦП в МК STM32F4

АЦП може функціонувати в режимі прямого доступу до пам'яті (*DMA*). Функція аналогового вартування (*analog watchdog*) дозволяє точно

контролювати перетворювану напругу одного, деяких або усіх обраних каналів. Коли перетворювана напруга знаходиться поза запрограмованих порогових значень, генерується переривання. Для синхронізації аналого-цифрового перетворення з таймерами АЦП можуть бути викликані будь-яким з таймерів *TIM1*, *TIM2*, *TIM3*, *TIM4*, *TIM5* або *TIM8*.

Температурний сенсор, показаний на рисунку 4.1, генерує напругу, яка змінюється лінійно зі збільшенням температури. Діапазон перетворення становить від 1,7 *В* до 3,6 *В*. Для перетворення вихідної напруги сигналу датчика температури в цифрове значення датчик внутрішньо підключений до вхідного каналу *ADC1_IN18*, тобто того ж, що використовується для вимірювання резервної напруги живлення МК *VBAT*. Коли перетворення датчика температури та *VBAT* включені одночасно, виконується перетворення тільки сигналу *VBAT*.

Внаслідок технологічного розкиду існує зсув напруг температурних датчиків від однієї до іншої ІС. Тому внутрішній датчик температури, в основному, підходить для застосувань, які виявляють зміни температури замість абсолютних температур. Якщо необхідне точне значення температури, має використовуватися зовнішній датчик.

АЦП підтримує такі режими перетворення: одноразове, безперервне, по триггеру, за таймером. Швидкість перетворення аналогового сигналу у цифровий код сягає 0,9 мільйонів вибірок за секунду з програмованим часом захоплення і перетворення. Також можлива автоматична калібровка модуля. Більш докладно з можливостями модуля АЦП можна ознайомитися в [7]. Взаємодія з модулем на рівні програми користувача ефективно реалізується за допомогою бібліотек *HAL* [16].

4.2 Особливості резидентних USART/UART в МК STM32F4

В МК вбудовувано чотири універсальні синхронні/асинхронні приймачі/передавачі (*USART1*, *USART2*, *USART3* і *USART6*) і чотири універсальні асинхронні приймачі/передавачі (*UART4*, *UART5*, *UART7* і *UART8*).

Ці інтерфейси забезпечують асинхронний зв'язок, підтримку *IrDA SIR ENDEC*, режим мультіпроцесорного зв'язку, режим однопровідного полудуплексного зв'язку, а також *LIN Master/Slave*. Інтерфейси *USART1* і *USART6* мають можливість спілкуватися зі швидкістю до 11,25 *Мбіт/с*, інші доступні інтерфейси працюють до 5,62 *Мбіт/с*.

USART1, *USART2*, *USART3* і *USART6* також забезпечують апаратне керування сигналами *CTS* і *RTS*, режим смарт-карт (сумісний з *ISO* 7816) і можливість *SPI*-подібного зв'язку. Всі інтерфейси можуть обслуговуватися контролером прямого доступу до пам’яті (*DMA*).

Порівняння можливостей різних модулів *USARTx* наведено в [7, 17]. Взаємодія з модулями на рівні програми користувача ефективно реалізується за допомогою бібліотек *HAL* [16].

4.3 Приклад використання АЦП та USART

Розглянемо процедуру вирішення наступної задачі. Зняти значення напруги на 15-ому вході *ADC*1 (*SB*2) із точністю до 4-ох знаків після коми. Результати аналого-цифрових перетворень за його закінченням треба перезаписувати в перериванні від АЦП у змінну типу `uint16_t`. За натисканням на кнопку *SB*1 знята напруга також має відсилатися до терміналу персонального комп'ютера за інтерфейсом *USART*6, що працює на частоті 9600 *бод*. Кнопку опитувати у обробнику переривань від таймера *TIM*7.

На рисунках 4.2 – 4.4 наведені схеми програмних блоків.

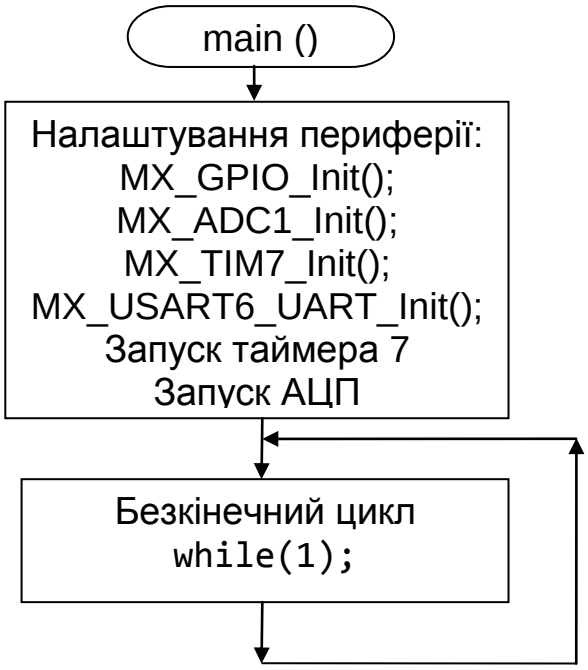


Рисунок 4.2 – Схема функції `main ()`

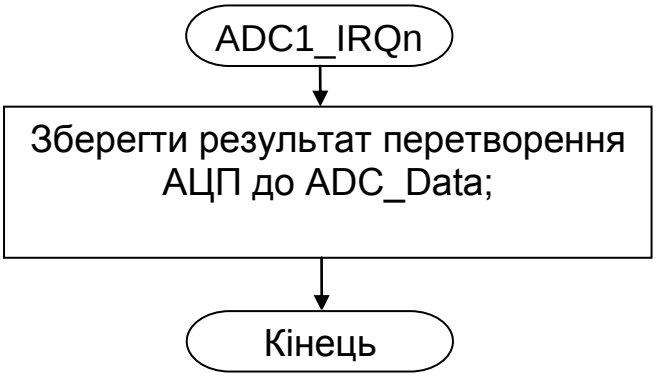


Рисунок 4.3 – Схема функції обробника переривання від ADC1

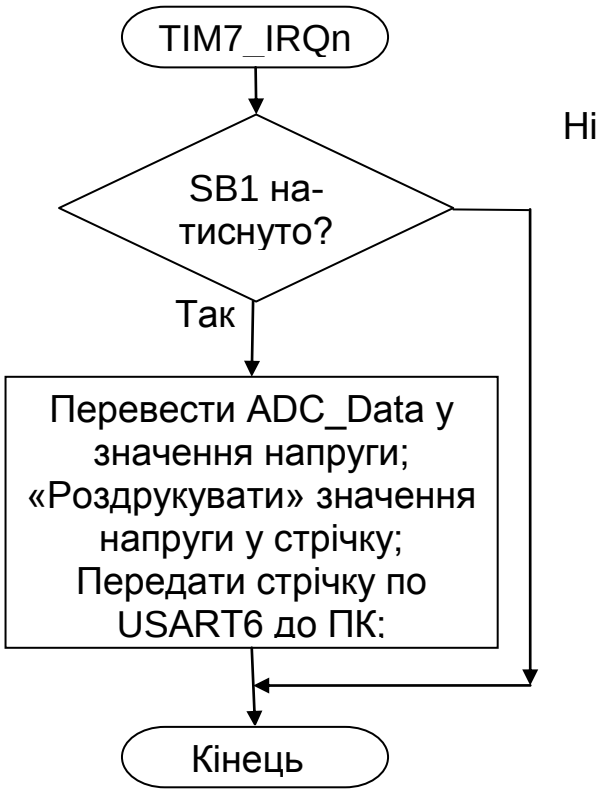


Рисунок 4.4 – Схема функції обробника переривання Таймера 7

4.3.1 Створення коду ініціалізації для МК STM32F429ZGT6

1. Створимо новий проект у програмі *STM32CubeMX* з назвою, наприклад, *MPT_Lab4*, і активуємо тактування від кварцового резонатора, обравши в групі RCC:

`High Speed Clock (HSE) = Crystal/Ceramic Resonator.`

2. Проведемо налаштування системи синхронізації МК на вкладці `Clock Configuration` аналогічно тому, як ми це робили раніше.

3. Налаштуємо роботу із кнопкою *SB*1, призначивши лінію порту `PC4`, як вхід загального призначення `GPIO`. На вкладці `Configuration` увімкнемо підтягуючий резистор (`GPIO Pull-Up Mode`) для `PC4`.

4. Активуємо 15-й вхід АЦП1 в якості зовнішнього переривання (рисунок 4.5).

5. Активуємо модуль `USART6` (рисунок 4.6), а також таймер `TIM7`.

6. Налаштуємо Таймер 7 на частоту виникнення переривання 100 *Гц* (рисунок 4.7), а також визначимо тип дії: `Update Event`. На вкладці `NVIC Settings` дозволимо переривання від `TIM7`.

7. Скоригуємо налаштування для `USART6` (рисунок 4.8).

8. Виконаємо конфігурацію модуля АЦП, увімкнувши для роботи регулярний канал (рисунок 4.9). Також увімкнемо переривання від АЦП на вкладці `NVIC Settings`.

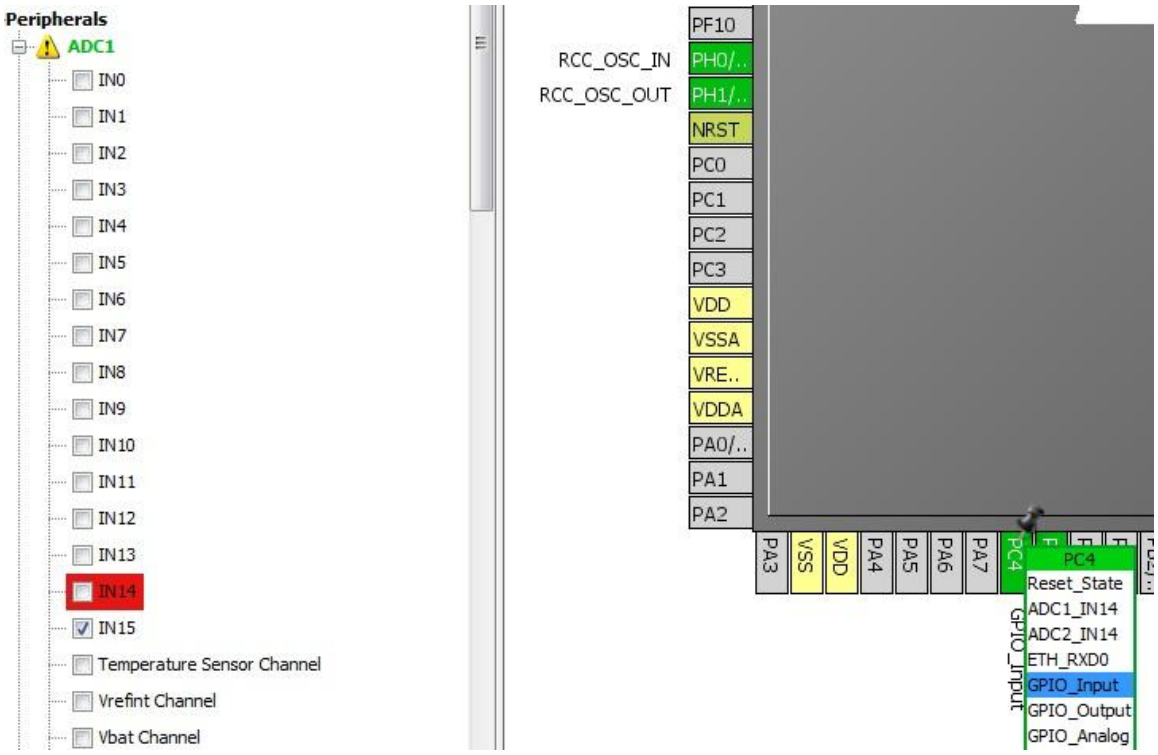


Рисунок 4.5 – Конфігурування виводів МК *STM32F429ZGT6* (*ADC*1)

9. Конфігурація налаштувань `NVIC` (рисунок 4.10) передбачає створення чотирьох обробників переривань, хоча `USART6 global interrupt` і не буде задіяне. Це потрібне для автоматичного створення відповідної змінної.

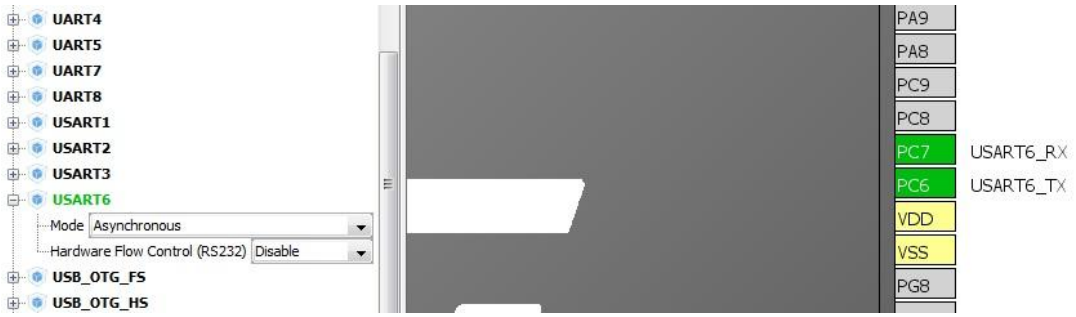


Рисунок 4.6 – Конфігурування виводів МК *STM32F429ZGT6* (*USART*6)

10. Генеруємо проект для `IDE Keil MDK-ARM V5`.

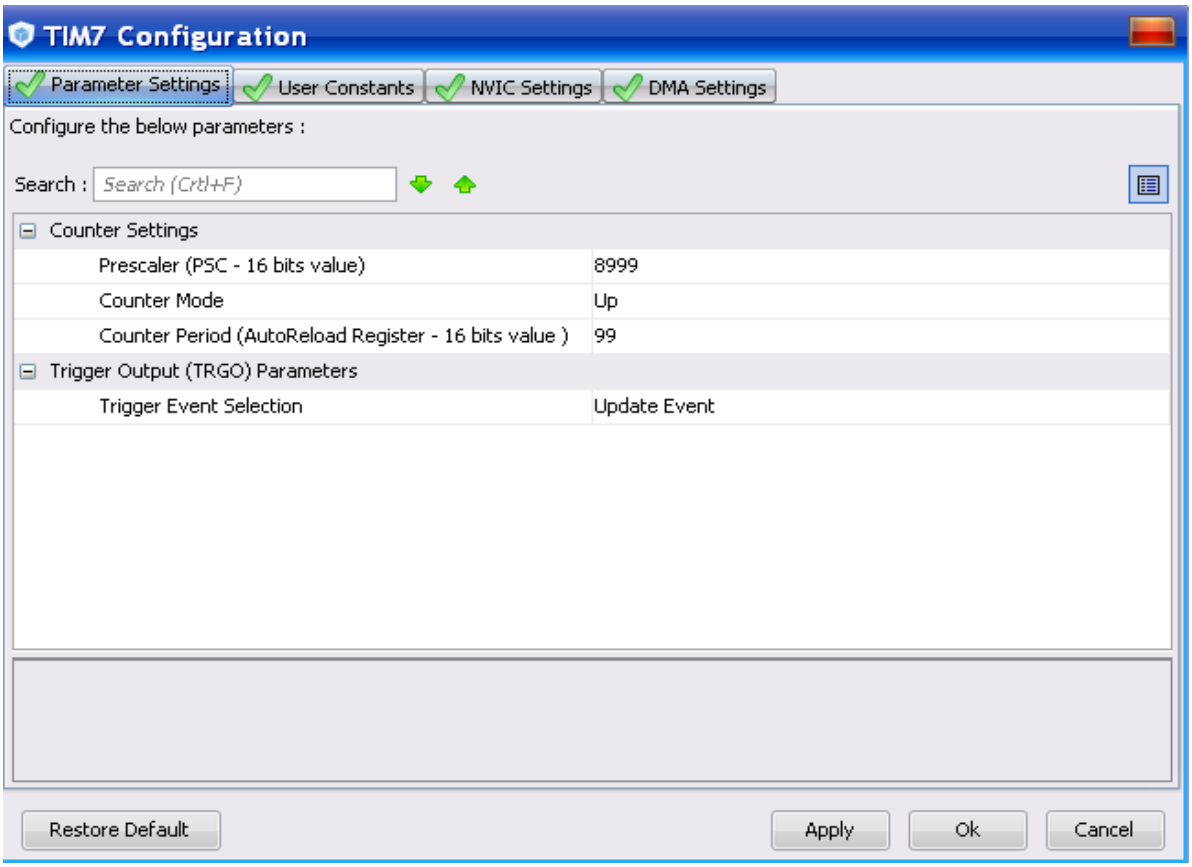


Рисунок 4.7 – Налаштування TIM7

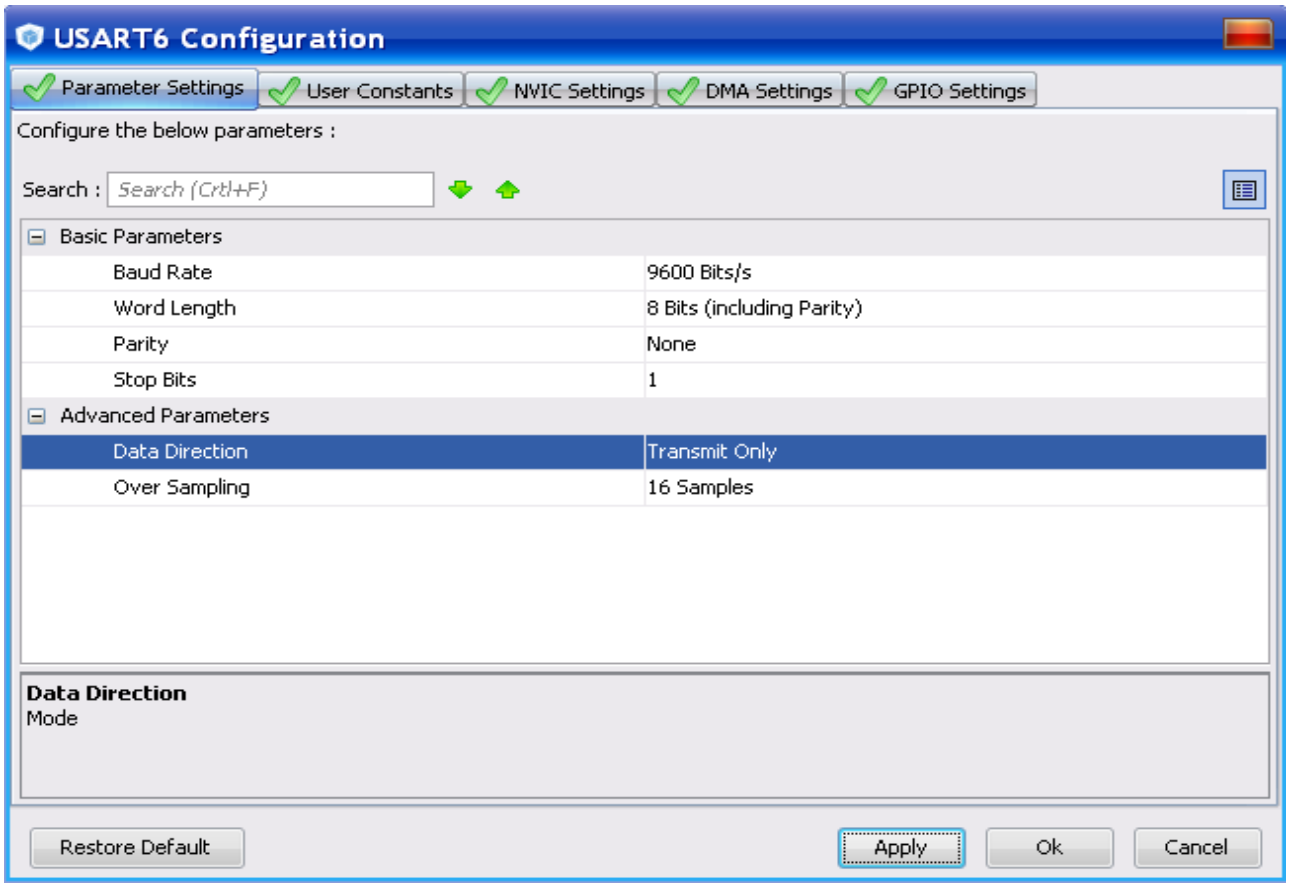


Рисунок 4.8 – Налаштування USART6

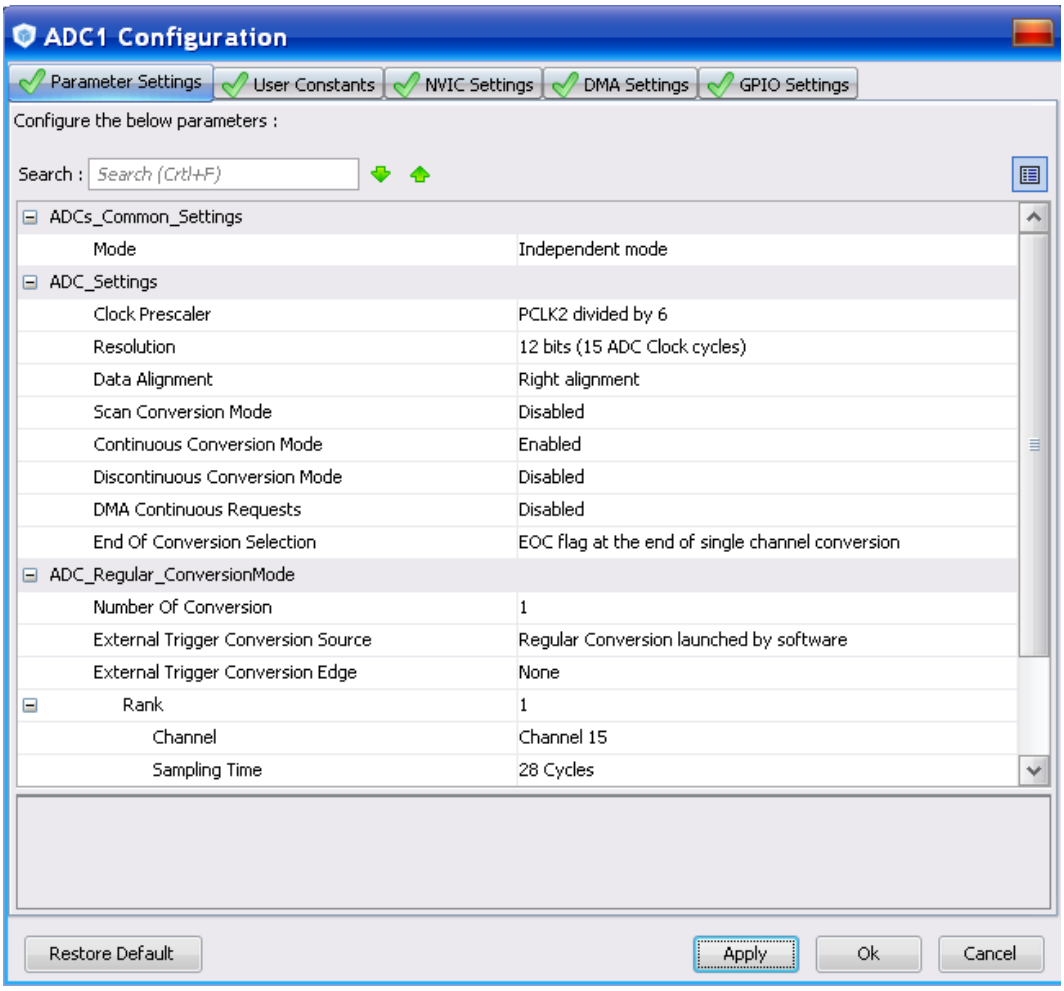


Рисунок 4.9 – Конфігурація налаштувань АЦП

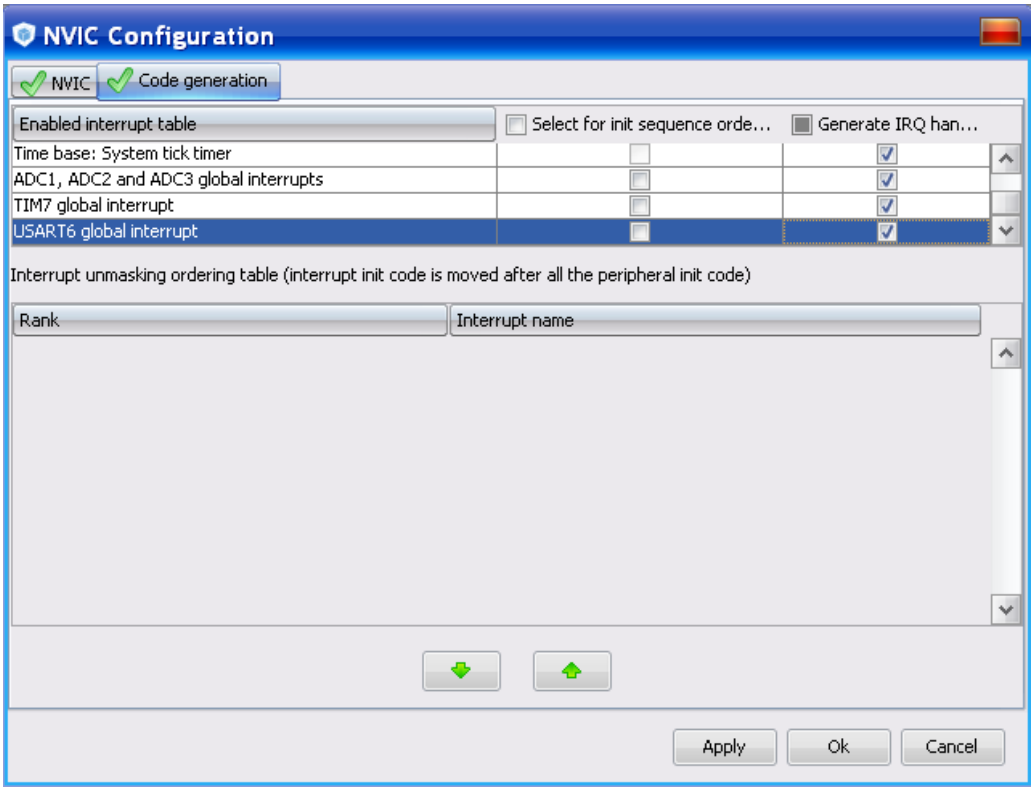


Рисунок 4.10 – Конфігурація налаштувань NVIC

4.3.2 Підготовка програми в IDE Keil MDK-ARM V5

1. Файл main.c

```
  /* USER CODE BEGIN 2 */
// Запускаємо таймер 7 і дозволяємо переривання
     HAL_TIM_Base_Start_IT(&htim7);
//Запускаємо АЦП і дозволяємо переривання
     HAL_ADC_Start_IT(&hadc1);
  /* USER CODE END 2 */
```

2. Файл stm32f4xx_it.c

```
/* USER CODE BEGIN 0 */
#include <string.h>
char str_Tx[80]; // Змінна для зберігання стрічок
volatile uint16_t ADC_Data; //Змінна для зберігання даних АЦП
volatile uint8_t push_count = 0; //Лічильник натискань кнопки
uint8_t USER_BUTTON_PUSH(void)//Функція опитування кнопки SB0
{
     if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_4) == GPIO_PIN_RESET)
     {push_count++;
          if(push_count > 15)
          {push_count = 0;    return 1; }
          else {              return 0; }
     } else {  push_count = 0;     return 0; }
}
/* USER CODE END 0 */

  /* USER CODE BEGIN TIM7_IRQn 1 */
     if(USER_BUTTON_PUSH())
     {
//Перетворення коду АЦП у напругу
float u = ((float)ADC_Data*3)/4096;
sprintf(str_Tx, "%.4fV\r\n", u); //"друк" стрічки
//передача стрічки по USART6
HAL_UART_Transmit(&huart6, (uint8_t*)str_Tx, strlen(str_Tx),
100);
     };
  /* USER CODE END TIM7_IRQn 1 */

/* USER CODE BEGIN 1 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc1)
{
ADC_Data = HAL_ADC_GetValue(hadc1); //Зчитування данних АЦП
HAL_ADC_Start_IT(hadc1); //Запуск переривання від АЦП
}
/* USER CODE END 1 */
```

4.3.3 Перевірка тестової програми

1. Запустіть термінальну програму та налаштуйте її. Покажемо це на прикладі стандартної програми `Hyper Terminal` з пакету операційної системи Windows.

`Программы -> Стандартные -> Связь -> Hyper Terminal`

2. Створіть нове підключення:

`Файл -> Новое подключение`

3. Задайте номер того порту, який запропонувала операційна система. Параметри підключення:

швидкість прийому 9600 бод;

посилка 8 біт;

без перевірки паритету;

з одним стоповим бітом (рисунок 4.11).

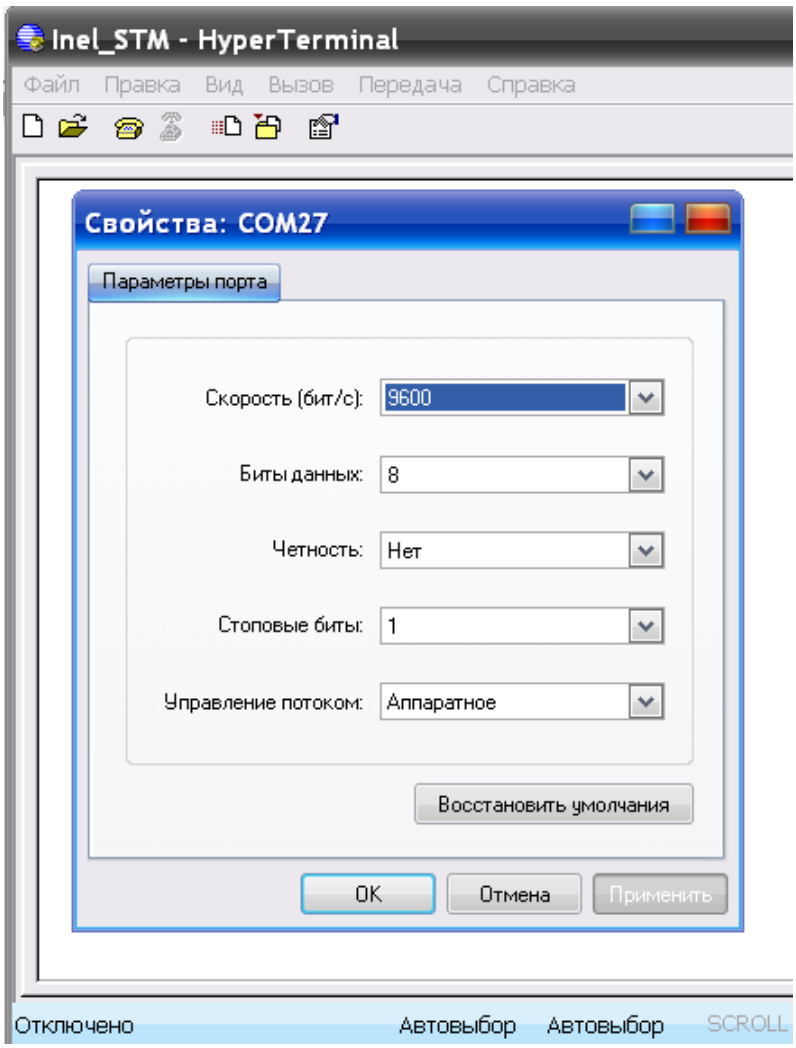


Рисунок 4.11 – Налаштування програми *Hyper Terminal*

4. Далі на стенді натискання кнопки *SB*1 має призводити до появи у вікні терміналу напруги, що вимірюється на вході *PC*5, тобто до якого підключена кнопка *SB*2.

5. За утримання кнопки *SB*2, маємо отримати результат 0 V.

4.4 Контрольні запитання

1. Наведіть методи, типи та алгоритми аналого-цифрового перетворення.

2. Дайте визначення наступних статичних параметрів АЦП: дискретизація, квантування, роздільна здатність, диференційна нелінійність, відхилення коефіцієнта перетворення, зсув нуля.

3. Дайте визначення наступних динамічних параметрів АЦП: час перетворення, час циклу перетворення, максимальна частота перетворення, апертурний час.

4. Перелічіть основні фактори, які впливають на похибку АЦП.

5. Який алгоритм АЦП реалізовано в МК *STM32F4*? Наведіть структуру ADC1 [7].

6. Які функції бібліотек *HAL* дозволяють взаємодіяти з модулем АЦП на рівні програми користувача? Наведіть порядок їхнього використання [16].

7. Які функції бібліотек *HAL* дозволяють взаємодіяти з модулем *UART* на рівні програми користувача? Наведіть порядок їхнього використання.

4.5 Хід роботи

1. Ознайомтеся з принципами побудови та функціонування модулів АЦП та *UART* в МК *STM32F4* [7, 16, 17].

2. Дайте письмові відповіді на контрольні запитання.

3. Безпосередньо у викладача отримайте допуск до виконання роботи в лабораторії та варіант завдання.

3. Утворіть новий проект (наприклад, `МПТ_lab4`), розробіть алгоритм і складіть програму відповідно до отриманого завдання.

5. Перевірте роботу розробленої програми в реальному часі на стенді *"Inel-STM"* та продемонструйте її викладачу.

6. Завершіть звіт та захистіть роботу.

4.6 Орієнтовні варіанти завдань

Знімати значення напруги із точністю до 4-ох знаків після коми на 15-ому вході *ADC*1. Результат перетворень АЦП перезаписувати у змінну типу uint16_t в перериванні від АЦП за закінченням перетворення. Старші 8 розрядів висвітити на одиничних індикаторах *HL*0…*HL*1.

1. За натисканням кнопки *SB*1 надсилати результат знятої напруги до терміналу ПК по інтерфейсу *USART*6, який повинен працювати на частоті 9600 *бод*. Кнопку опитувати у обробнику переривань від таймера *TIM*6.

2. Відсилати результат знятої напруги до терміналу ПК у перериванні від таймера *TIM*6 (частота переповнення – 1 *Гц*) по інтерфейсу *USART*6, який повинен працювати на частоті 115200 *бод*.

3. Відсилати результат знятої напруги до терміналу ПК у відповідь на відправку до МК цифри «2» по інтерфейсу *USART*6, який повинен пра-

цювати на частоті 9600 *бод*.

4. Відсилати середнє значення десяти результатів знятої напруги до терміналу ПК у перериванні від таймера *TIM*7 (частота переповнення – 0,5 *Гц*) по інтерфейсу *USART*6, який повинен працювати на частоті 57600 *бод*.

5. Відсилати 5 результатів знятої напруги за натисканням кнопки *SB*1 до терміналу ПК у перериванні від таймера *TIM*6 (частота переповнення – 1 *Гц*) по інтерфейсу *USART*6, який повинен працювати на частоті 115200 *бод*. Кнопку опитувати у обробнику переривань від таймера *TIM*7.

4.7 Вимоги до звіту по роботі

Звіт повинен містити функціональну схему підключення застосованих апаратних засобів до МК з вказівкою конкретних виводів, завдання, алгоритми і коментовані тексти розроблених програм. Необхідно також навести опис функцій та їхніх параметрів, блоків даних, зробити висновки щодо досягнутої мети роботи.

5 Лабораторна робота №5. Дослідження методів виведення аналогової інформації в МПС на базі STM32F4

Мета роботи: навчитися формувати аналогові електричні сигнали та організовувати автономну сумісну взаємодію з контролером прямого доступу до пам'яті для створення системи із мінімальним використанням процесорного часу ARM-МК STM32F429ZGT6.

5.1 Особливості цифро-аналогових перетворювачів (ЦАП) в ARM-МК STM32F4

Два 12-бітних буферизованих канали ЦАП *DAC1* та *DAC2* можна використати для перетворення двох цифрових послідовностей у аналогові сигнали напруги на виходах *DAC_OUT1* та *DAC_OUT2* (рисунок 5.1).

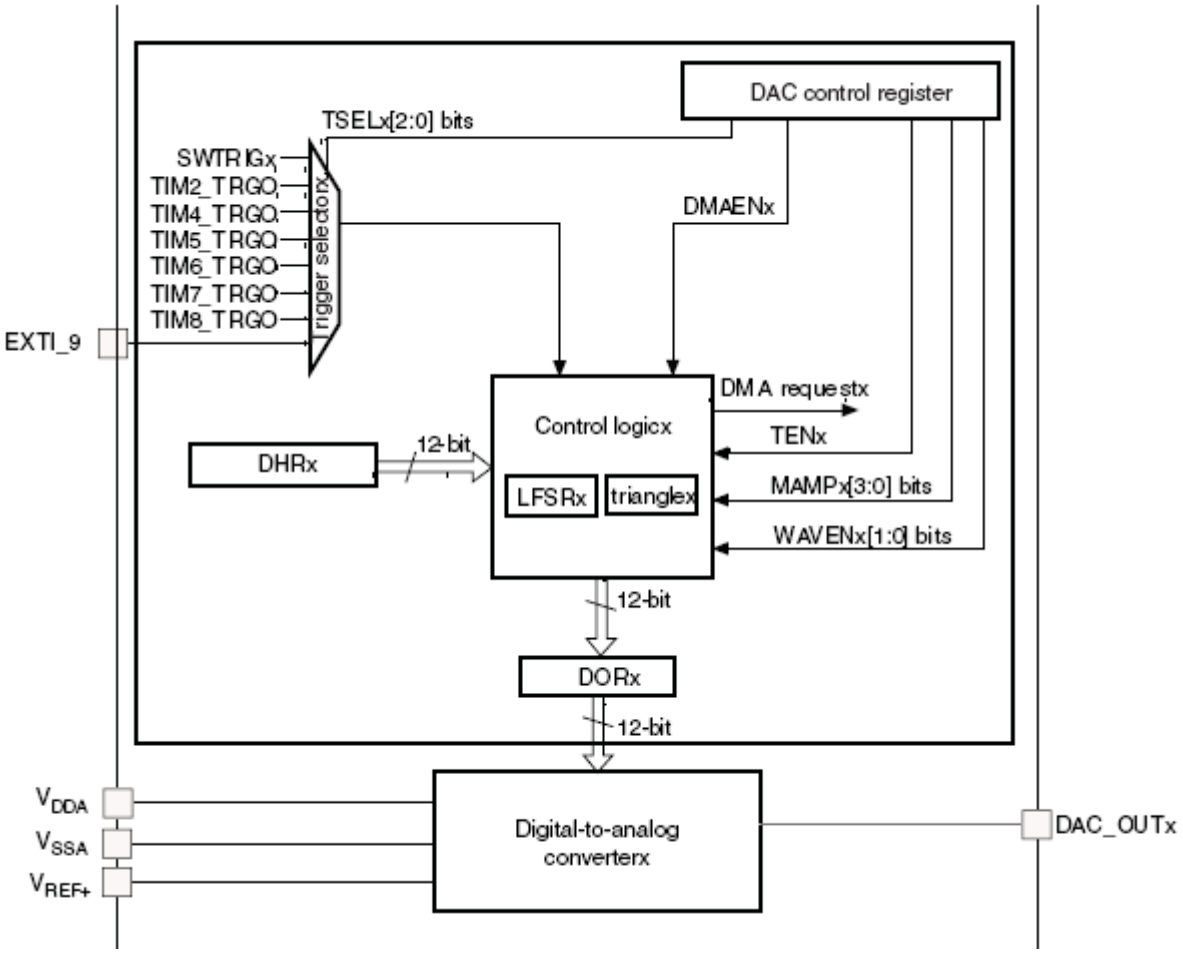


Рисунок 5.1 – Структурна схема каналу x DAC (x=1/2)

На рисунку застосовані наступні позначення:

TENx (trigger enable) – біт дозволу роботи каналу x DAC. Цей біт встановлюється і скидається за допомогою програмного забезпечення, щоб увімкнути/вимкнути можливість запуску ЦАП каналу x.

SWTRIGx (SoftWare TRIGger control bit) – біт програмного запуску АЦП.

Якщо біт керування TENx встановлено, перетворення може бути викликано зовнішньою подією (таймер, зовнішня лінія переривання).

TIMy_TRGO (Timer y TRGO event) – сигнали запуску ЦАП від внутрішньокристальних таймерів МК. y = 2, 4, 5, 6, 7, 8.

TSELx[2:0]: біти вибору запуску DAC каналу x.

DMAENx – біт дозволу роботи DAC каналу x в режимі DMA. Цей біт встановлюється і скидається за допомогою програмного забезпечення.

DHRx (data holding register) – регістр утримання даних каналу x. Саме сюди треба записати дані, які, за певних умов, можуть попасти на ЦАП.

LFSRx (linear feedback shift register) – регістр зсуву з лінійним зворотним зв'язком. Використовується для генерації псевдошумового сигналу із змінною амплітудою,

MAMPx[3:0] – біти вибору маски/амплітуди DAC каналу x. Ці біти записуються за допомогою програмного забезпечення для вибору маски або амплітуди в режимі генерації трикутного періодичного сигналу.

WAVENx[1:0] – біти дозволу генерації шуму/трикутного періодичного сигналу DAC каналу x. Ці біти встановлюються/скидаються програмно.

DORx (data output register) регістр вихідних даних каналу x.

DAC_OUTx. ЦАП використовує два виводи МК, а саме – PA4 і PA5, які у разі налаштування ЦАП починають виконувати альтернативну функцію виходу аналогового сигналу *DAC_OUT1*та *DAC_OUT2*, відповідно.

Даний подвійний цифровий інтерфейс підтримує такі функції:

• 8- або 12-розрядний вихід;
• можливість синхронізувати оновлення виходів;
• генерація шумоподібного сигналу;
• формування трикутного періодичного сигналу;
• незалежне або синхронне перетворення на двох каналах ЦАП;
• можливість прямого доступу до пам’яті для кожного каналу;
• зовнішні тригери для запуску перетворення.

В пристрої використовуються вісім входів тригерів ЦАП. Канали ЦАП ініціюються через виходи поновлення таймера, які також підключені до різних потоків DMA.

Цифрові входи перетворюються у вихідні напруги в діапазоні між 0 і V_{REF+}. Аналогові вихідні напруги на виході кожного з каналів ЦАП визначаються наступним рівнянням:

$$DAC_{output} = V_{REF+} \times DOR/4095,$$

де DOR – вміст вихідного регістру ЦАП;

V_{REF+} – вхід опорної напруги.

5.1.1 Загальні (generic) API-інтерфейси HAL для ЦАП

Цифро-аналогове перетворення може бути викликаним:

1) Зовнішньою подією: EXTI Line 9 (будь-який GPIOx_Pin9) з використанням опції DAC_TRIGGER_EXT_IT9. Використана ніжка МК (GPIOx_Pin9) має бути налаштована в режимі введення.

2) TRGO таймерів: TIM2, TIM4, TIM5, TIM6, TIM7 і TIM8 (опції DAC_TRIGGER_T2_TRGO, DAC_TRIGGER_T4_TRGO …).

3) Програмним забезпеченням з використанням опції DAC_TRIGGER_SOFTWARE.

Формат даних ЦАП може бути:

1) 8-біт з вирівнюванням вправо за допомогою DAC_ALIGN_8B_R.

2) 12-біт з вирівнюванням по лівому краю: DAC_ALIGN_12B_L.

3) 12-біт з вирівнюванням по правому краю: DAC_ALIGN_12B_R.

Порядок використання HAL-драйверів для ЦАП:

1) Щоб отримати доступ на запис до регістрів DAC з використанням функції HAL_DAC_Init (), має бути включений тактовий сигнал DAC APB.

2) Налаштувати DAC_OUTx (DAC_OUT1: PA4, DAC_OUT2: PA5) на роботу в аналоговому режимі.

3) Налаштувати канали ЦАП з використанням функції HAL_DAC_ConfigChannel ().

4) Дозволити роботу каналу ЦАП з використанням функції HAL_DAC_Start () або HAL_DAC_Start_DMA.

Робота АЦП в режимі опитування:

1) Запустіть ЦАП, використовуючи HAL_DAC_Start ().

2) Щоб прочитати останнє значення вихідних даних ЦАП, використовуйте функцію HAL_DAC_GetValue ().

3) Зупиніть ЦАП, використовуючи HAL_DAC_Stop ().

Робота АЦП в режимі прямого доступу до пам'яті (DMA):

1) Запустіть ЦАП з використанням HAL_DAC_Start_DMA (). На даному етапі користувач повинен вказати довжину даних, що мають передаватися щоразу наприкінці перетворення.

2) В кінці передачі даних виконується функція HAL_DAC_ConvCpltCallbackCh1 () або …Ch2 (), і користувач може додати свій власний код за допомогою настройки покажчика функції HAL_DAC_ConvCpltCallbackCh1 або …Ch2.

3) У разі помилкової передачі виконується функція HAL_DAC_ErrorCallbackCh1 () або …Ch2 (), і користувач може додати свій власний код за допомогою настройки покажчика функції HAL_DAC_ErrorCallbackCh1 або …Ch2 ().

4) Зупиніть ЦАП з використанням HAL_DAC_Stop_DMA ().

Найбільш часто використовувані макроси в HAL-драйвері ЦАП:

1) __HAL_DAC_ENABLE: включити ЦАП;

2) __HAL_DAC_DISABLE: відключити ЦАП;

3) __HAL_DAC_CLEAR_FLAG: очистити прапори очікування ЦАП.

4) __HAL_DAC_GET_FLAG: отримати прапор стану обраного ЦАП.

З іншими корисними макросами HAL-драйверу ЦАП можна ознайомитися у файлі заголовка.

Для ініціалізації і деініціалізації ЦАП використовують функції:

HAL_DAC_Init()

HAL_DAC_DeInit()

HAL_DAC_MspInit()
HAL_DAC_MspDeInit()

Функції введення/виведення ЦАП представлені наступними інтерфейсами:

HAL_DAC_Start()
HAL_DAC_Stop()
HAL_DAC_Start_DMA()
HAL_DAC_Stop_DMA()
HAL_DAC_GetValue()
HAL_DAC_IRQHandler()
HAL_DAC_ConvCpltCallbackCh1()
HAL_DAC_ConvHalfCpltCallbackCh1()
HAL_DAC_ErrorCallbackCh1()
HAL_DAC_DMAUnderrunCallbackCh1()

Для управління периферією ЦАП використовують функції:

HAL_DAC_ConfigChannel()
HAL_DAC_SetValue()

Для стану і помилок ЦАП використовують наступні інтерфейси:

HAL_DAC_GetState()
HAL_DAC_GetError()
HAL_DAC_IRQHandler()
HAL_DAC_ConvCpltCallbackCh1()
HAL_DAC_ConvHalfCpltCallbackCh1()
HAL_DAC_ErrorCallbackCh1()
HAL_DAC_DMAUnderrunCallbackCh1()

Більш докладну інформацію по кожній generic-функції API-інтерфейсу HAL можна знайти в [16].

5.1.2 Розширений HAL DAC драйвер

Коли включений подвійний режим (тобто канали 1 та 2 DAC працюють одночасно), використовують:

1) HAL_DACEx_DualGetValue (), щоб отримати цифрові дані, які будуть перетворені;

2) HAL_DACEx_DualSetValue (), щоб встановити цифрові значення для одночасного перетворювалися в каналах 1 і 2;

3) HAL_DACEx_TriangleWaveGenerate () для генерації трикутного сигналу;

4) HAL_DACEx_NoiseWaveGenerate () для генерації шумового сигналу.

Подробиці використання даних та інших функцій ЦАП можна знайти в [16].

5.2 Особливості контролера прямого доступу до пам'яті (DMA) в ARM-МК STM32F4

Прямий доступ до пам'яті (DMA), використовується для того, щоб забезпечити високошвидкісну передачу даних між периферійними пристроями і пам'яттю, а також між пам'яттю та пам'яттю. За допомогою DMA можна швидко перемістити дані без будь-яких дій процесора. Це зберігає ресурси процесора вільними для інших операцій.

Щоб оптимізувати пропускну здатність системи, заснованої на складній шинній матричній архітектурі, контролер DMA поєднує в собі потужну подвійну ведучу AHB-шину з незалежним стеком FIFO.

Два контролери DMA підтримують по 8 потоків (в цілому – 16). Кожен з контролерів призначений для управління запитами доступу до пам'яті від одного або декількох периферійних пристроїв. Кожен потік може мати до 8 каналів (запитів). І кожен з них має арбітра для обробки пріоритету між запитами DMA.

Основні особливості DMA:

• Подвійна ведуча AHB-шина, одна з яких призначена для доступу до пам'яті, а інша – для периферійних доступів.

• Підлеглий інтерфейс AHB підтримує тільки 32-розрядні доступи.

• 8 потоків для кожного контролера DMA, до 8 запитів на потік.

• Для кожного потоку є буфери пам'яті глибиною чотири 32-розрядні слова. Їх можна використати в режимі FIFO або в прямому режимі. В режимі FIFO програмно вибирається пороговий рівень 1/4, 1/2 або 3/4 від розміру FIFO.

Кожен запит DMA негайно ініціює передачу з/в пам'ять. Коли DMA налаштований на роботу в прямому режимі (FIFO відключений), для передачі даних в режимі з пам'яті до периферії попередньо з пам'яті у внутрішній стек FIFO завантажуються тільки одні дані. Як тільки за сигналом периферійного пристрою спрацьовує запит DMA, це забезпечує негайну передачу даних.

• Кожен потік можна налаштовати за допомогою апаратних засобів:

– як регулярний канал, який підтримує пересилання периферія-пам'ять, пам'ять-периферія і пам'ять-пам'ять;

– як канал з подвійним буфером, який також підтримує подвійну буферізацію на стороні пам'яті.

• Кожен з 8 потоків підключений до виділених апаратних каналів DMA (запитів).

• Пріоритети між запитами потоків DMA керовані програмно (4 рівня, які складаються з дуже високого, високого, середнього, низького) або апаратно в разі рівності (запит 0 має пріоритет відносно запиту 1 тощо).

• Кожен потік також підтримує програмний запуск (тригер) для передачі з пам'яті до пам'яті (доступно тільки для контролера DMA2).

• Кожен запит потоку може бути вибраний з 8 можливих запитів каналу. Цей вибір настроюється програмно і дозволяє декільком периферій-

ним пристроям ініціювати запити на DMA.

• Кількість елементів даних, що підлягають передачі, може управлятися або контролером DMA, або периферійним пристроєм:

– контролер потоку DMA: кількість елементів даних, що підлягають передачі, задаються програмно в діапазоні від 1 до 65535;

– периферійний контролер потоку: кількість елементів даних, що підлягають передачі, невідома і управляється периферійним пристроєм джерела або призначення, які сигналізують про кінець передачі за допомогою апаратних засобів.

• Незалежна ширина передачі джерела і призначення (байт, півслово, слово). Коли ширина даних джерела і призначення не рівні, DMA автоматично упаковує/розпаковує необхідні пересилання для оптимізації пропускної здатності. Ця функція доступна тільки в режимі FIFO.

• Інкрементна або неінкрементна адресація джерела і призначення.

• Підтримка додаткових пакетів передач з 4, 8 або 16 посилок. Розмір пакета задається програмно і, як правило, дорівнює половині розміру FIFO.

• Кожен потік підтримує управління кільцевим буфером.

• П’ять прапорів подій (половина передачі DMA, передачу DMA завершено, помилка передачі DMA, помилка FIFO DMA, помилка прямого режиму). Прапори об’єднані логічним АБО разом в одному запиті переривання для кожного потоку.

5.2.1 Програмно-апаратний драйвер API DMA

Використання драйвера:

1) дозволити та настроїти периферійний пристрій, що буде підключений до потоку DMA (за винятком внутрішньої пам'яті SRAM/FLASH: не потребує ініціалізації). Подробиці – у Довідковому керівництві з зв'язку між периферійними пристроями та запитами DMA;

2) для даного потоку запрограмувати необхідну конфігурацію за допомогою наступних параметрів: напрям передачі, формати даних джерела і призначення, режим управління (круговий, нормальний або периферійний потік), рівень пріоритету потоку, режим інкременту джерела і призначення, режим FIFO і його граничне значення (якщо це необхідно), пакетний режим для джерела і/або призначення (при необхідності) за допомогою функції `HAL_DMA_Init ()`.

Робота в режимі опитування (поллінгу):

1) після конфігурації адрес джерела і призначення і довжини переданих даних використайте `HAL_DMA_Start ()`, щоб почати передачу DMA;

2) використайте `HAL_DMA_PollForTransfer ()` для опитування кінця поточного сеансу передачі даних. В цьому випадку користувач в залежності від застосування може налаштувати фіксований тайм-аут.

Робота в режимі переривання:

1) налаштуйте пріоритет переривань DMA з використанням `HAL_NVIC_SetPriority ()`;

2) включіть обробник DMA IRQ за допомогою `HAL_NVIC_EnableIRQ ();`

3) після конфігурації адреси джерела, адреси призначення і довжини переданих даних використайте `HAL_DMA_Start_IT ()`, щоб почати передачу DMA. В цьому випадку конфігурується переривання DMA;

4) використайте виклик `HAL_DMA_IRQHandler ()` з підпрограми переривання `DMA_IRQHandler ()`.

В кінці передачі даних виконується функція `HAL_DMA_IRQHandler ()`, і користувач може додати свою власну функцію шляхом настройки покажчика функції `XferCpltCallback` і `XferErrorCallback` (тобто елемент структури обробника DMA).

Подробиці використання даних та інших функцій драйвера ПДП можна знайти в [16].

5.2.2 Розширений HAL DMA драйвер

Можливості розширених функцій:

1) налаштування адрес джерела і одержувача та довжини даних, а також запуск багатобуферної передачі DMA;

2) налаштування адрес джерела і одержувача та довжини даних, а також запуск багатобуферної передачі DMA з перериванням;

3) зміна адреси memory0 або memory1 під час роботи.

Включено наступні API:

1) `HAL_DMAEx_MultiBufferStart();`

2) `HAL_DMAEx_MultiBufferStart_IT();`

3) `HAL_DMAEx_ChangeMemory()`.

Розширений драйвер HAL DMA можна використати в такий спосіб. Почати багатобуферну передачу за допомогою функції `HAL_DMA_MultiBufferStart ()` для режиму опитування або `HAL_DMA_MultiBufferStart_IT ()` для режиму переривання. У режимі передачі пам'ять-пам'ять режим `Multi (Double) Buffer` не допускається. Коли включений режим `Multi (Double) Buffer`, передача циклічна за замовчуванням. В режимі `Multi (Double) Buffer` можна оновити базову адресу порту пам'яті AHB на льоту (`DMA_SxM0AR` або `DMA_SxM1AR`), коли потік включений.

5.3 Приклад спільного використання ЦАП та ПДП

Розглянемо процедуру вирішення наступної задачі. Треба за натисканням на кнопку *SB*1 почати виводити на ніжку МК DAC_OUT2 сигнал синусоїдальної форми. За повторним натисканням виведення сигналу треба припинити. Масив відліків синуса складається з 256 12-розрядних значень. ЦАП тактується від таймеру *TIM6* (частота роботи таймера – 1,125 *МГц*) і отримує відліки сигналу за допомогою DMA. Кнопку опитувати у перериванні від таймеру *TIM7*.

На рисунках 5.2 та 5.3 наведені схеми програмних блоків.

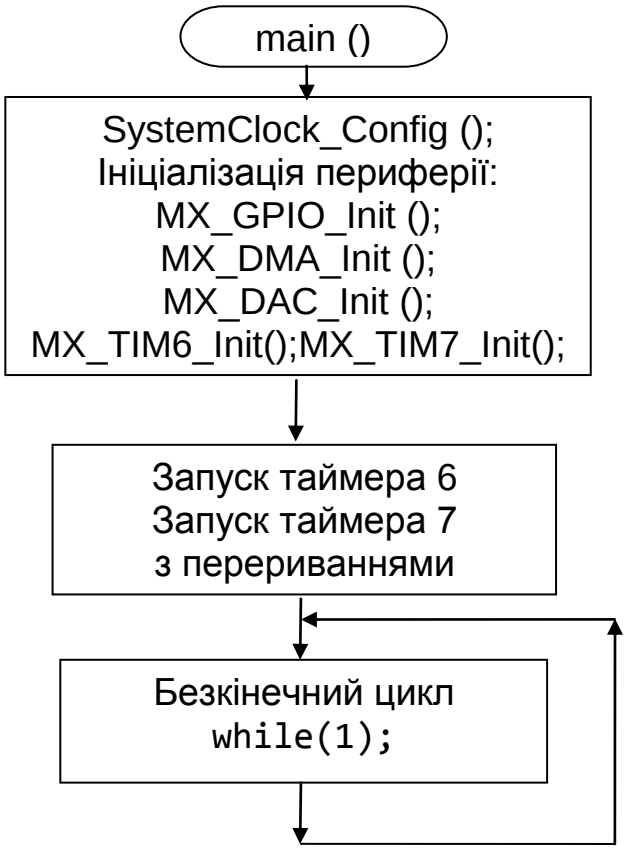


Рисунок 5.2 – Схема функції `main ()`

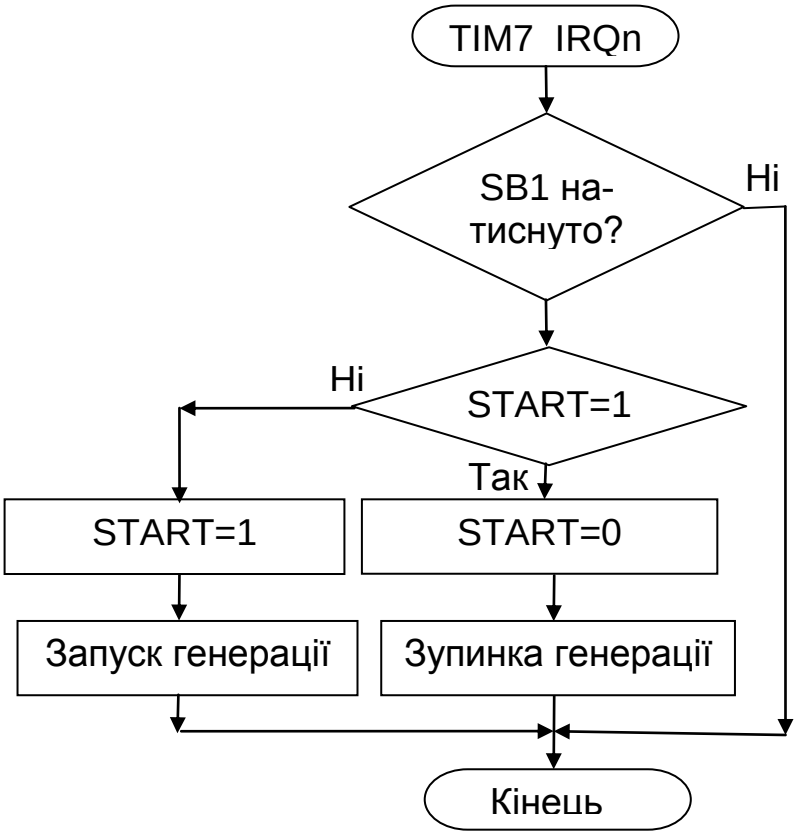


Рисунок 5.3 – Схема функції обробника переривання Таймера 7

5.3.1 Створення коду ініціалізації для МК STM32F429ZGT6

1. Створимо новий проект у програмі *STM32CubeMX* з назвою, наприклад, *MPT_Lab5* і активуємо тактування від кварцового резонатора, обравши в групі RCC:

`High Speed Clock (HSE) = Crystal/Ceramic Resonator.`

2. Проведемо налаштування системи синхронізації МК на вкладці `Clock Configuration` аналогічно тому, як ми це робили раніше.

3. Налаштуємо роботу із кнопкою *SB*1, призначивши лінію порту `PC4`, як вхід загального призначення `GPIO`. На вкладці `Configuration` увімкнемо підтягуючий резистор (`GPIO Pull-Up Mode`) для `PC4`.

4. Коніфгурування ЦАП та виводу `PA5` (*DAC*2) проводимо, як показано на рисунку 5.4.

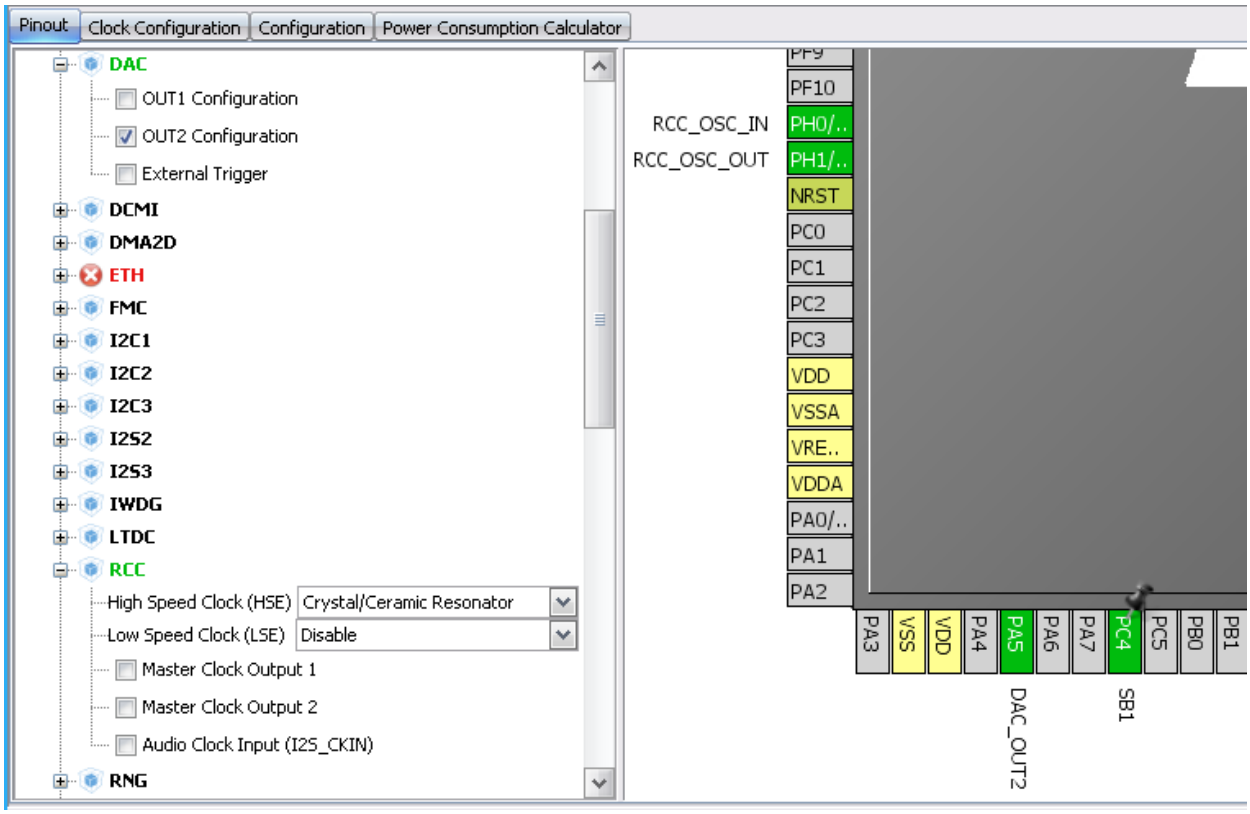


Рисунок 5.4 – Конфігурування виводу МК *STM32F429ZGT6* (*DAC*2)

5. Активуємо таймери 6 та 7. Налаштування таймерів `TIM6` і `TIM7` на вкладці `Configuration` показано на рисунках 5.5 і 5.6, відповідно. Для обох таймерів увімкнемо переривання.

6. Налаштування модуля ЦАП показано на рисунку 5.7.

7. Налаштування модуля ПДП показано на рисунку 5.8.

8. Генеруємо проект для `IDE Keil MDK-ARM V5`.

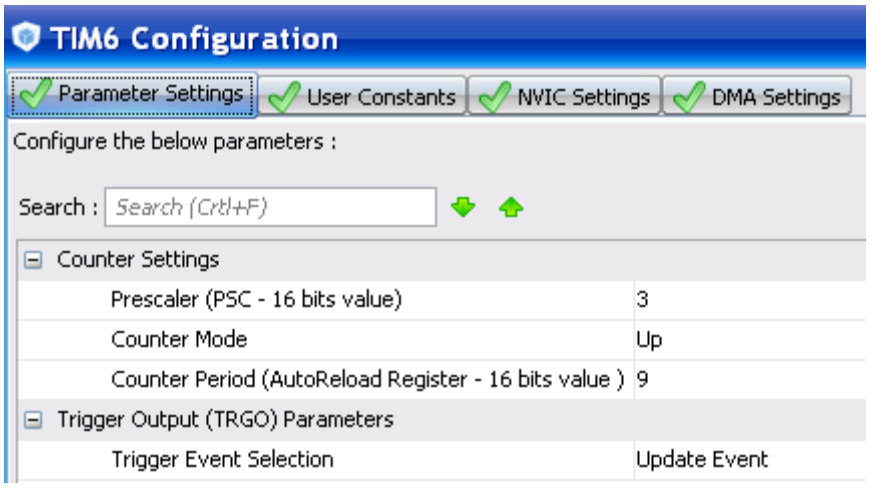


Рисунок 5.5 – Налаштування таймера TIM6

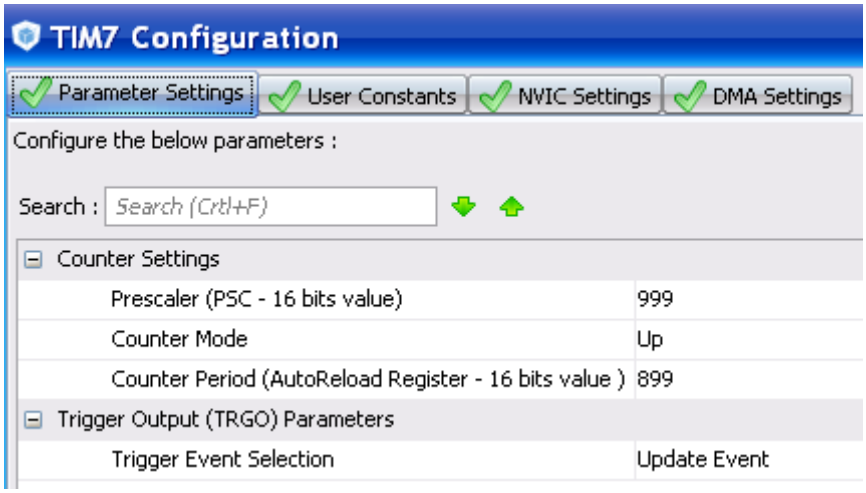


Рисунок 5.6 – Налаштування таймера TIM7

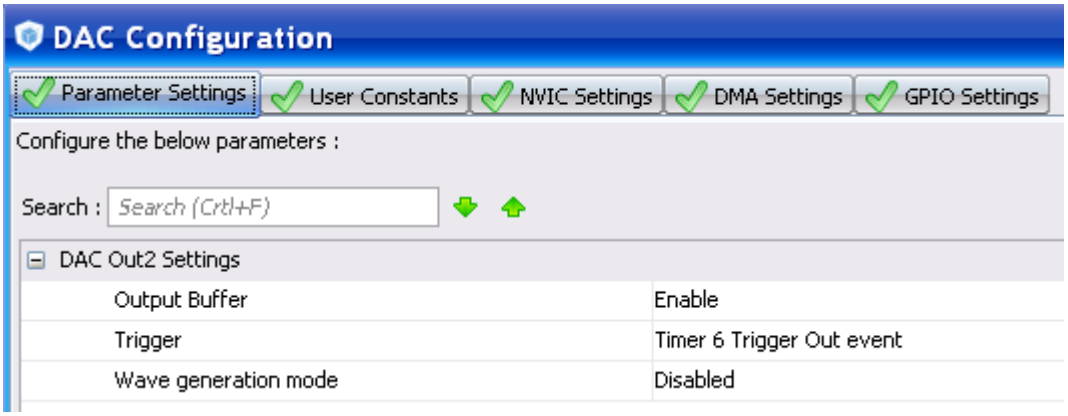


Рисунок 5.7 – Налаштування модуля ЦАП

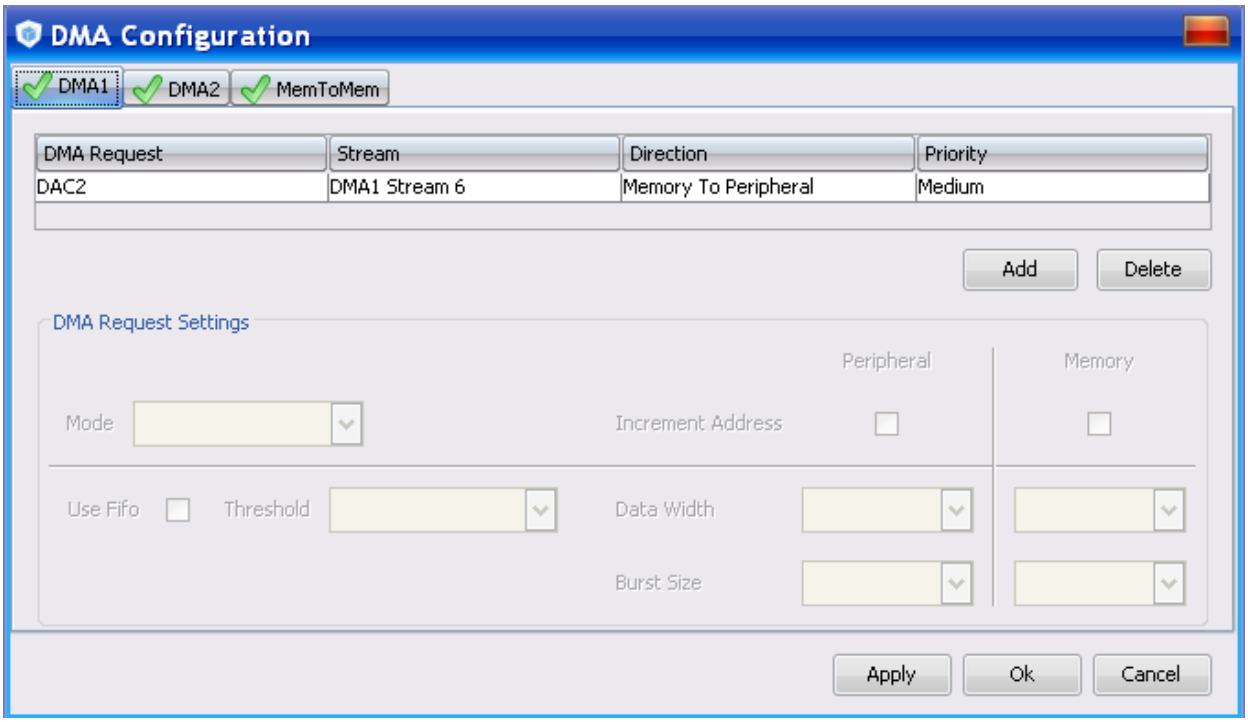


Рисунок 5.8 – Налаштування модуля ПДП

5.3.2 Підготовка програми в IDE Keil MDK-ARM V5

1. Файл main.c

```
/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------*/
const uint16_t sine_table[256] =
{
2048,2098,2148,2198,2248,2298,2348,2398,2447,2496,2545,2594,
     /* Тут наведений фрагмент таблиці значень синуса */
1797,1847,1897,1947,1997
};
/* USER CODE END PFP */

  /* USER CODE BEGIN 2 */
     HAL_TIM_Base_Start(&htim6);
     HAL_TIM_Base_Start_IT(&htim7);
  /* USER CODE END 2 */
```

2. Файл stm32f4xx_it.c

```
/* USER CODE BEGIN 0 */
extern const uint16_t sine_table[256];
volatile _Bool START = 0;
volatile uint8_t push_count = 0; //лічильник натискань кнопки
uint8_t USER_BUTTON_PUSH(void)//функція опитування кнопки SB0
{    if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_4) == GPIO_PIN_RESET)
     {    push_count++;
          if(push_count > 15) { push_count = 0;return 1; }
```

```
          else { return 0;    }
     } else {  push_count = 0;     return 0; }
}
/* USER CODE END 0 */

  /* USER CODE BEGIN TIM7_IRQn 1 */
     if(USER_BUTTON_PUSH())
     {
          if(START == 1)
          {
               START = 0;
               HAL_DAC_Stop_DMA(&hdac, DAC_CHANNEL_2);
          } else {
               START = 1;
               HAL_DAC_Start_DMA(&hdac, DAC_CHANNEL_2,
(uint32_t*)sine_table, 256, DAC_ALIGN_12B_R);
          }
     }
  /* USER CODE END TIM7_IRQn 1 */
```

5.4 Контрольні запитання

1. Наведіть методи, типи та алгоритми цифро-аналогового перетворення.
2. Перелічіть основні фактори, що впливають на похибку ЦАП.
3. Які функції бібліотек HAL дозволяють взаємодіяти з модулем ЦАП на рівні програми користувача? Наведіть порядок їхнього використання.
4. Визначте поняття ПДП. Опишіть структуру ПДП в ARM-МК STM32F4.
5. Яким чином можна розрахувати рівень аналогової напруги на виході ЦАП?
6. Які функції бібліотек HAL дозволяють периферії МК взаємодіяти з пам’яттю в режимі ПДП? Наведіть порядок їхнього використання.
7. Поясніть призначення окремих рядків програми прикладу 5.3.

5.5 Хід роботи

1. Вивчіть особливості цифро-аналогових перетворювачів (ЦАП) та контролера прямого доступу до пам’яті (DMA) в ARM-МК STM32F4.
2. Оформіть першу частину звіту з виконання лабораторної роботи, в якій дайте письмові відповіді на контрольні запитання (п. 5.4).
3. Безпосередньо у викладача отримайте допуск до виконання роботи в лабораторії та варіант завдання.
4. Розробіть алгоритм, складіть програму (згідно з варіантом), перевірте її роботу програми на стенді і продемонструйте викладачеві.

5.6 Орієнтовні варіанти завдань для розробки програми

1. За натисканням кнопки *SB*1 розпочати генерацію сигналу трикутної форми на виводі DAC_OUT2, по наступному натиску генерація припиняється. Масив відліків трикутника складається з 128 12-розрядних значень. ЦАП тактується від таймеру TIM6 (частота роботи таймера – 1 МГц) і отримує відліки сигналу за допомогою DMA. Кнопку опитувати у перериванні від таймеру TIM6.

2. За натисканням кнопки *SB*1 починається/припиняється генерація сигналу пилкоподібної форми на виводі DAC_OUT1. Масив відліків пилкоподібного сигналу складається з 64 восьмирозрядних значень. ЦАП працює у 8-розрядному режимі, тактується від таймера TIM6 (частота роботи 1МГц) і отримує відліки сигналу за допомогою DMA. Кнопку опитувати у перериванні від таймера TIM6.

3. За натисканням кнопки *SB*2 починається/припиняється генерація сигналу прямокутної форми на виводі DAC_OUT2. Масив відліків трикутника складається з 256 12-розрядних значень. ЦАП тактується від таймера TIM6 (частота роботи таймера – 0,5МГц) і отримує відліки сигналу за допомогою DMA. Кнопку опитувати у перериванні від таймера TIM7.

4. За натисканням кнопки *SB*1 починається/припиняється генерація синусоїдального сигналу на виводі DAC_OUT1. Масив відліків трикутника складається із 512 восьмирозрядних значень. ЦАП працює у 8-розрядному режимі, тактується від таймера TIM6 (частота роботи таймера – 1МГц) і отримує відліки сигналу за допомогою DMA. Кнопку опитувати у перериванні від таймера TIM4.

5. За натисканням кнопки *SB*1 починається/припиняється генерація синусоїдального сигналу на виводі DAC_OUT1. За натисканням кнопки *SB*2 починається/припиняється генерація косинусоїдального сигналу на виводі DAC_OUT1.

5.7 Вимоги до звіту по роботі

Звіт повинен містити функціональну схему підключення застосованих апаратних засобів до МК з вказівкою конкретних виводів, завдання, алгоритми і коментовані тексти розроблених програм. Необхідно також навести опис функцій та їхніх параметрів, блоків даних, зробити висновки щодо досягнутої мети роботи.

6 Лабораторна робота №6. Дослідження інтегрованого середовища розробки програмного забезпечення IAR Embedded Workbench

Мета роботи: дослідити особливості встановлення та можливості IAR Embedded Workbench під час налагодження та виконання програм.

Заснована у Швеції у 1983 році, *IAR Systems* є виробником поширених у світі програмних інструментів для розробки вбудованих систем на основі 8-, 16- і 32-розрядних процесорів. Це продукти, в основному, в галузі автомобілебудування, побутової електроніки, промислової автоматизації, медичних приладів та телекомунікацій.

IAR Embedded Workbench є повністю інтегрованим середовищем розробки, яке включає компілятор, асемблер, компонувальник і відладчик. Один звичний для розробника набір інструментів допомагає забезпечити безперервний робочий процес з різними МК. Цей набір включає в себе:

IDE TOOLS (інструменти інтегрованого середовища розробки):
Editor (редактор вхідних текстів);
Project manager (менеджер проектів);
Library tools (бібліотечні інструменти).

BUILD TOOLS (інструменти побудови):
IAR C/C++ Compiler (компілятор IAR C/C++);
Assembler (асемблер);
Linker (редактор зв'язків).

C-SPY DEBUGGER (відладчик C-SPY):
Simulator (симулятор);
Hardware debugging (апаратний відладчик);
Power debugging (відладчик енергоспоживання);
RTOS plugins (компоненти RTOS).

6.1 Особливості встановлення IAR Embedded Workbench

1. На сайті виробника [18] у вікні `Showing tools for…` обрати `ARM`.

2. Далі треба знайти опис *IAR embedded workbench for ARM* і, після ознайомлення з продуктом, перейти до розділу *Download a free trial*.

Оціночна ліцензія повністю безкоштовна і дозволяє спробувати інтегроване середовище розробки та оцінити його ефективність та особливості використання. При запуску продукту в перший раз буде запропоновано зареєструватися, щоб отримати оціночну ліцензію.

3. Після установки є такі варіанти на вибір:

– 30-денна обмежена за часом, але повністю функціональна ліцензія;
– обмежена за розміром коду ліцензія *Kickstart* без будь-якого обме-

ження за часом.

Обмеження за 30-денною ліцензією:

- не надається вихідний код бібліотеки часу виконання;
- немає підтримки MISRA C;
- відлагодження вхідного коду мовою Сі C-RUN обмежується у розмірі 12 Кбайт за винятком постійних даних;
- обмежена технічна підтримка;
- не повинна використовуватися для будь-якого комерційного застосування.

Обмеження оціночної ліцензії *Kickstart*:

- обмеження коду 32 Кбайт (16 Кбайт для Cortex-M0/M0+/M1);
- не надається вихідний код бібліотеки часу виконання;
- немає підтримки MISRA C;
- C-RUN не доступне;
- обмежена технічна підтримка.

4. Отже, натиснувши кнопку *Download Software*, ми розпочнемо завантаження поточної версії *IAR Embedded Workbench for ARM* (під час написання даного навчального видання це *Version 7.70, 1049.31 MB*). Далі треба інсталювати застосування, погодившись, зокрема, на встановлення драйвера *ST-Link*, аби надалі завантаження флеш-пам'яті програм МК можна було б здійснювати безпосередньо з інтегрованого середовища відлагодження ПЗ.

5. Після першого запуску *IAR Embedded Workbench for ARM* з'явиться вікно майстра ліцензування (`License Wizard`), в якому треба відмітити `Register with IAR Systems to get an evaluating license`, а далі у наступному вікні натиснути кнопку `Register`. У вікні реєстрації треба вибрати тип ліцензії (наприклад, **`Code size limited`** `IAR Embedded Workbench for ARM, v. 7.70, 32K Kickstart Edition`) вказати всі необхідні (позначені *) дані і, нарешті, натиснути `Submit Registration`. На вказану у реєстраційній анкеті адресу електронної пошти надійде повідомлення, у якому буде розміщене пряме посилання на підтвердження реєстрації. Перейшовши за цим посиланням, Ви отримаєте номер ліцензії (`license number`), який треба ввести (або просто скопіювати та вставити) у порожнє поле вікна реєстрації майстра ліцензування (рисунок 6.1).

6. В результаті Ви отримаєте деталі підтвердженої ліцензії, наприклад, як на рисунку 6.2). Після подальшого переходу можна отримати повідомлення про те, що ліцензію активовано (наприклад, `Your permanent license has been activated`), а також – про тривалість дії цієї ліцензії (наприклад, `Your product is now ready for permanent use`). Натиснувши кнопку `Done`, Ви завершите процес ліцензування і зможете переходити безпосередньо до роботи з *IAR Embedded Workbench for ARM*.

Рисунок 6.1 – Вставляння номеру ліцензії

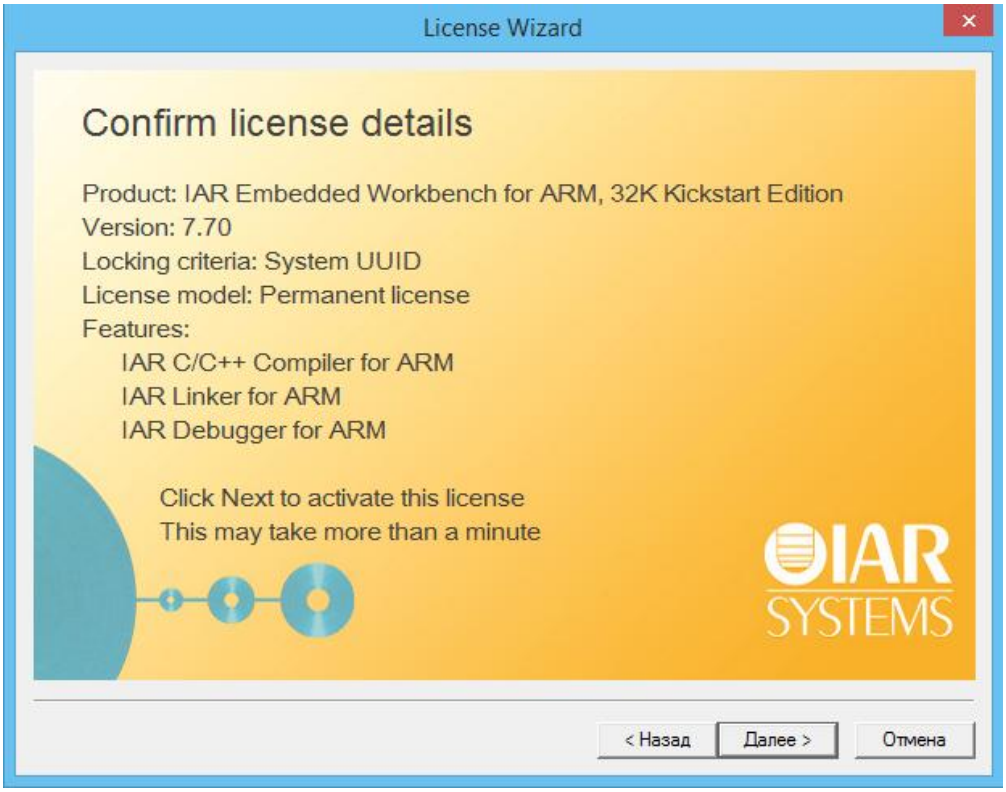


Рисунок 6.2 – Деталі підтвердженої ліцензії

6.2 Створення коду ініціалізації для МК STM32F429ZGT6

Покроково розглянемо вирішення задачі програмування МК *STM32F429ZGT6* в середовищі *IAR Embedded Workbench for ARM*. Завдання повністю повторює те, що було розглянуте в підрозділі 2.3.

1. Використаємо схему програми, зображену на рисунку 2.8.

2. Запускаємо програму *STM32CubeMX* і послідовно виконуємо дії з настроювання МК, описані у пунктах 2 та 3 підрозділу 2.3.

3. На вкладці Project->Settings… (Alt+P) вводимо шлях для зберігання проекту, називаємо проект і обираємо IDE (у нашому випадку – IWARM). Натискаємо Ok.

4. Тепер можна генерувати код ініціалізації: Project->Generate Code (Ctrl+Shift+G).

5. Після генерації проекту можна одразу ж перейти до його відкриття в IDE *IAR Embedded Workbench*, обравши Open Project.

6.3 Відлагодження програми в IDE *IAR Embedded Workbench*

1. Після відкриття згенерованого проекту на екрані відео монітору з'являється так званий робочій простір (Workspace) *IDE IAR Embedded Workbench*.

2. Файли користувача у дереві проекту знаходяться у групі "Application/User". В інших групах можна знайти файли бібліотеки HAL та драйвери CMSIS.

3. Обов'язково треба дослідити опції проекту Project->Options (Alt+F7). У категорії Debugger на вкладці Setup у віконному меню, що випадає, можна обрати, зокрема, ST-LINK.

4. За відсутності апаратних засобів тут же треба вибрати Simulator.

5. Додайте рядки програми до файлу main.c аналогічно підрозділу 2.4.

6. Для перевірки працездатності проекту скомпілюємо його, натиснувши F7 (Project->Make).

7. Для завантаження програми до резидентної флеш-пам'яті та початку першого відлагодження треба скористатися Project->Download and Debug (Ctrl+D).

8. Для повторного відлагодження програми без завантаження до резидентної флеш-пам'яті треба скористатися Project-> Debug without Downloading, що продовжить життєвий цикл МК.

9. Процес відлагодження передбачає крокування за програмою (меню Debug), спостереження за ресурсами МК та програми (регістри, змінні, меню View) тощо.

6.4 Контрольні запитання

1. З яких компонентів складається набір *IAR Embedded Workbench*?

2. Які основні відмінності файлової структури *IDE IAR Embedded Workbench* проекту від *Keil® MDK*?

3. Що дозволяють настроювати опції проекту?

4. Як в робочому просторі вивести вміст файлу з описом МК?

5. Як швидко проект *IAR Embedded Workbench* перетворити в проект *Keil® MDK*?

6.5 Хід роботи

1. Виконайте дії, описані в підрозділах 6.1, 6.2 та 6.3.

2. Дослідіть структуру робочого простору та пункти меню *IDE IAR Embedded Workbench*.

3. Дослідіть опції проекту *IDE IAR Embedded Workbench*.

4. Оформіть першу частину звіту з виконання лабораторної роботи, в якій зазначте прізвища студентів бригади та дайте письмові відповіді на контрольні запитання (п. 6.4).

5. Безпосередньо у викладача отримайте допуск до виконання роботи в лабораторії та варіант завдання.

6. Розробіть алгоритм, складіть програму (згідно з варіантом), перевірте роботу програми на стенді і продемонструйте викладачеві.

7. Завершіть звіт та захистіть роботу.

6.6 Орієнтовні варіанти завдань

1. Відповідають підрозділу 2.8.
2. Комплексний проект, який містить роботу з кнопками, одиничними індикаторами

6.7 Вимоги до звіту по роботі

Звіт повинен містити функціональну схему підключення застосованих апаратних засобів до МК з вказівкою конкретних виводів, завдання, алгоритми і коментовані тексти розроблених програм. Необхідно також навести опис функцій та їхніх параметрів, блоків даних, зробити висновки щодо досягнутої мети роботи.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Cortex-M4 Processor [Електронний ресурс]. – Режим доступу : https://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php

2. Войтенко В., Білорус І. Навчальний лабораторний стенд Inel-STM // Технічні науки та технології : науковий журнал / Черніг. нац. технол. ун-т. – Чернігів : Черніг. нац. технол. ун-т, 2016. – № 3 (5). – С. 131–138.

3. ARM® Cortex®-M4 Processor Technical Reference Manual. Revision r0p1 [Електронний ресурс]. – Режим доступу : http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.100166_0001_00_en/index.html

4. STM32F429ZG. High-performance advanced line, ARM Cortex-M4 core with DSP and FPU, 1 Mbyte Flash, 180 MHz CPU, ART Accelerator, Chrom-ART Accelerator, FMC with SDRAM, TFT. [Електронний ресурс]. – Режим доступу : https://my.st.com/content/my_st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32f4-series/stm32f429-439/stm32f429zg.html

5. STM32Cube initialization code generator [Електронний ресурс]. – Режим доступу : http://www.st.com/en/development-tools/stm32cubemx.html

6. ST-Link, ST-Link/V2, ST-Link/V2-1 USB driver signed for XP, Windows7, Windows8. [Електронний ресурс]. – Режим доступу : https://my.st.com/content/my_st_com/en/products/embedded-software/development-tool-software/stsw-link009.html

7. RM0090. STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM®-based 32-bit MCUs. Reference manual. DocID018909 Rev 12. – STMicroelectronics, 2016. – 1744 p.

8. PM0214. STM32F3, STM32F4 and STM32L4 Series Cortex®-M4 programming manual. DocID022708 Rev 5. – STMicroelectronics, 2016. – 260 p.

9. MDK Microcontroller Development Kit [Електронний ресурс]. – Режим доступу : http://www2.keil.com/mdk5/

10. ARMbed. Platforms. [Електронний ресурс]. – Режим доступу : https://developer.mbed.org/platforms/

11. ARM Microcontroller Development Kit. [Електронний ресурс]. – Режим доступу : https://www.keil.com/demo/eval/arm.htm#/DOWNLOAD

12. STMicroelectronics STM32F429ZGTx. [Електронний ресурс]. – Режим доступу : http://www.keil.com/dd2/st/stm32f429zgtx

13. CMSIS – Cortex Microcontroller Software Interface Standard. [Електронний ресурс]. – Режим доступу : http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php

14. MDK v4 Legacy Support. [Електронний ресурс]. – Режим доступу : http://www2.keil.com/mdk5/legacy

15. CP210x USB to UART Bridge VCP Driver. [Електронний ресурс]. –

Режим доступу : https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

16. UM 1725. Description of STM32F4xx HAL drivers. User Manual DOCID025834 Rev 3. – STMicroelectronics, 2015. – 963 p. [Електронний ресурс]. – Режим доступу : https://my.st.com/content/my_st_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32cube-embedded-software/stm32cubef4.html

17. STM32F427xx, STM32F429xx Datasheet – production data. DocID024030 Rev 8. – STMicroelectronics, 2016. – 233 p.

18. IAR Embedded Workbench. [Електронний ресурс]. – Режим доступу : https://www.iar.com/iar-embedded-workbench/#!?architecture=ARM

Додаток А
Елементи принципової схеми *„Inel-CBI”*

А.1 Лінійка з 8 світлодіодів на платі процесора

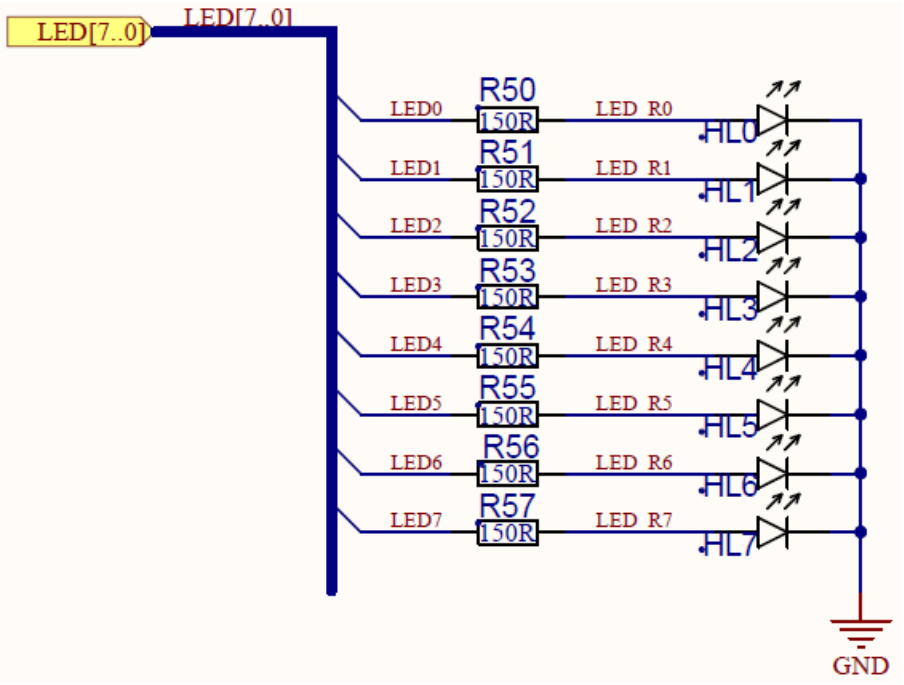


Рисунок А.1 – Схема підключення одиничних індикаторів в стенді

А.2 Підключення кнопок в стенді

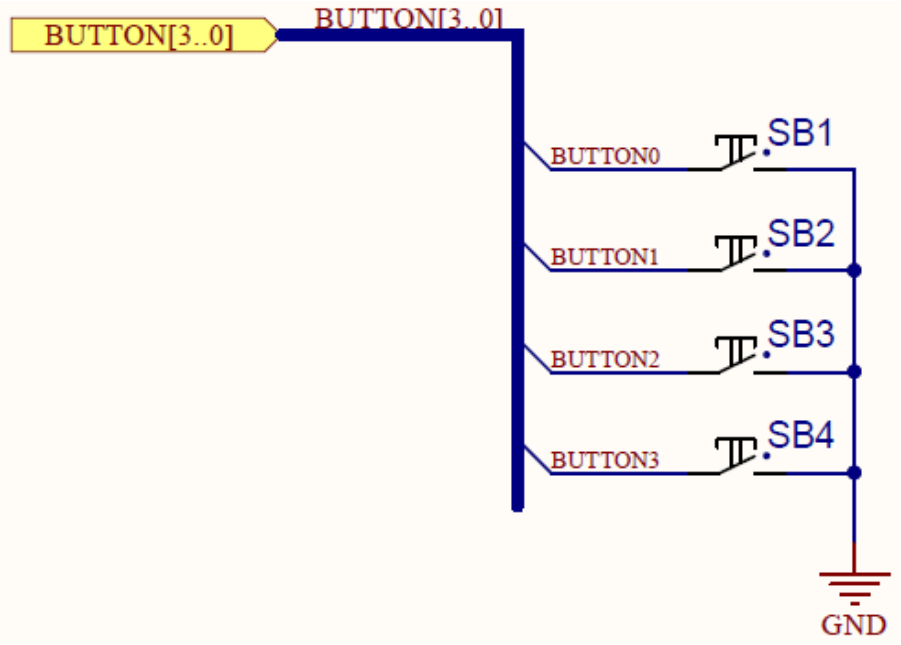


Рисунок А.2 – Схема підключення кнопок в стенді

А.3 Роз'єми розширення на платі процесора

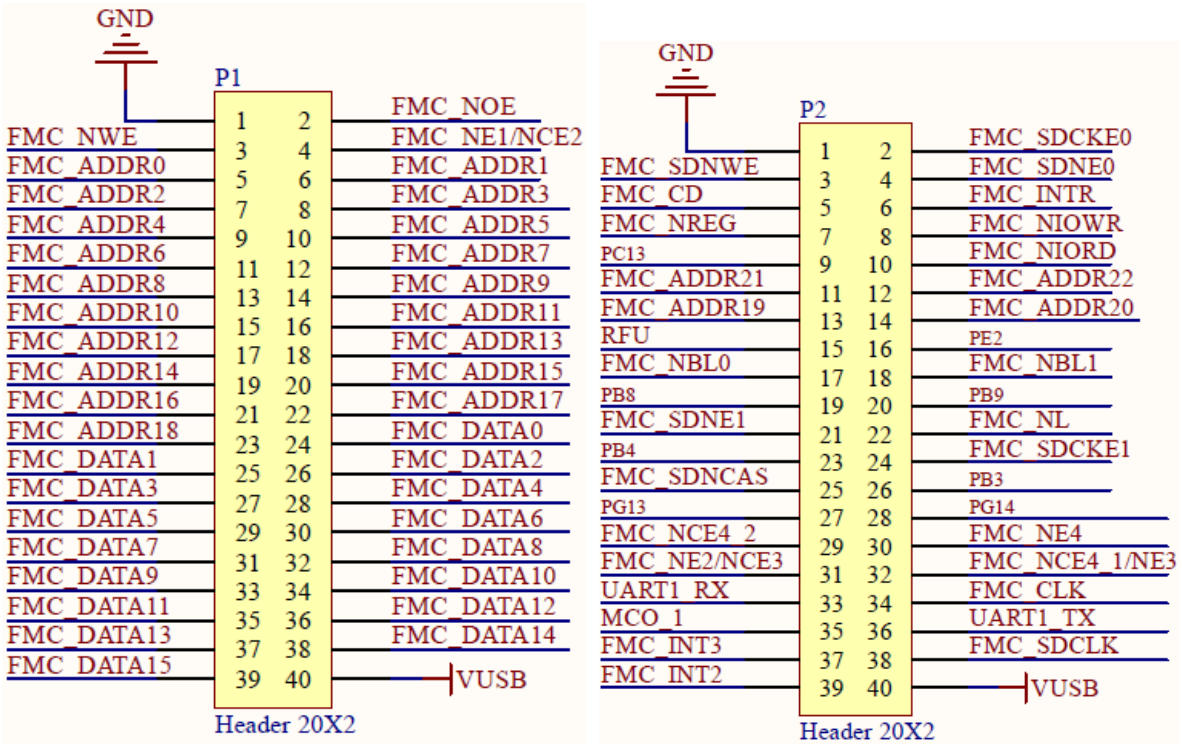


Рисунок А.3 – Схема роз'ємів розширення в стенді

Таблиця А.1 – Сигнали на роз'ємах системного інтерфейсу стенду Inel-STM

X1		X2	
1	GND	1	GND
2	PD4	2	PC3
3	PD5	3	PC0
4	PD7	4	PC2
5	PF0	5	PF9
6	PF1	6	PF10
7	PF2	7	PF7
8	PF3	8	PF8
9	PF4	9	PC13
10	PF5	10	PF6
11	PF12	11	PE5
12	PF13	12	PE6
13	PF14	13	PE3
14	PF15	14	PE4
15	PG0	15	RFU
16	PG1	16	PE2
17	PG2	17	PE0
18	PG3	18	PE1
19	PG4	19	PB8
20	PG5	20	PB9

X1		X2	
21	PD11	21	PB6
22	PD12	22	PB7
23	PD13	23	PB4
24	PD14	24	PB5
25	PD15	25	PG15
26	PD0	26	PB3
27	PD1	27	PG13
28	PE7	28	PC14
29	PE8	29	PG11
30	PE9	30	PG12
31	PE10	31	PG9
32	PE11	32	PG10
33	PE12	33	PA10
34	PE13	34	PD3
35	PE14	35	PA8
36	PE15	36	PA9
37	PD8	37	PG7
38	PD9	38	PG8
39	PD10	39	PG6
40	VUSB	40	VUSB