Лабораторна робота № 3

Вивчення GPIO мікроконтролерів PSoC 6 фірми Cypress

Мета роботи: Ознайомитися з системою вводу/виводу (GPIO) мікроконтролерів PSoC 6 фірми Cypress; методами налаштування, читання, запису виводів портів мікроконтролера.

Теоретичні відомості

Система вводу/виводу забезпечує інтерфейс між ядром центрального процесора (ЦП) та периферійними компонентами для зовнішнього світу. Гнучкість мікроконтролерів PSoC 6 і можливість їх вводів/виводів значно спрощує проектування схем та компоновку компонент схеми на платі. Виводи GPIO в сімействі мікроконтролерів PSoC 6 об'єднані в порти. Порт може мати максимум вісім виводів GPIO.

Основні характеристики GPIO мікроконтролерів PSoC 6:

- можливість вводу і виводу аналогових та цифрових сигналів;
- вісім режимів роботи виводів портів;
- окремі регістри для читання порту і запису в порт;
- виводи з захистом від допустимих перевищень напруги (OVT-GPIO);
- окремі джерела вводу/виводу і напруг для до шести груп вводу/виводу;
- можливі переривання по передньому фронту, по задньому фронту або по обох фронтах на всіх GPIO;
- контроль швидкості наростання фронтів сигналів виводів GPIO;
- режим утримання для фіксації попереднього стану (використовується для зберігання стану вводу/виводу в режимі глибокого сну);
- вибір режиму CMOS або низьковольтного вхідного буферу з LVTTL;
- підтримка сенсору дотику (CapSense);
- дають можливість виконувати логічні функції в колі вводу/виводу з інтелектуальними вводами/виводами;
- підтримка сегменту LCD пристрою.

Мікроконтролер PSoC 6 оснащений аналоговою і цифровою периферією. На рис. 3.1 зображено вигляд з'єднань між периферійними пристроями та виводами.

Виводи GPIO під'єднані до комірок вводу/виводу. До складу цих комірок входять вхідний буфер для цифрового входу, який забезпечує високий вхідний опір і драйвери для цифрових вхідних сигналів. Цифрова периферія під'єднується до комірок вводу/виводу через високошвидкісну матрицю вводу/виводу (HSIOM). Матриця HSIOM для кожного виводу містить мультиплексори з допомогою яких під'єднують вибрані периферійні пристрої з виводом. Матриця HSIOM також є мостом між цифровим системним з'єднанням (digital system interface - DSI) і виводами. Це дозволяє направляти сигнали з виводів на цифрову периферійну мережу UDB, під'єднану до DSI. Аналогові периферійні пристрої, такі як SAR ADC, компаратор з низьким споживанням струму (LPCOMP) і блоком неперервного часу (CTB), сенсор дотику CapSense, під'єднуються до виводів GPIO напряму або через AMUXBUS.

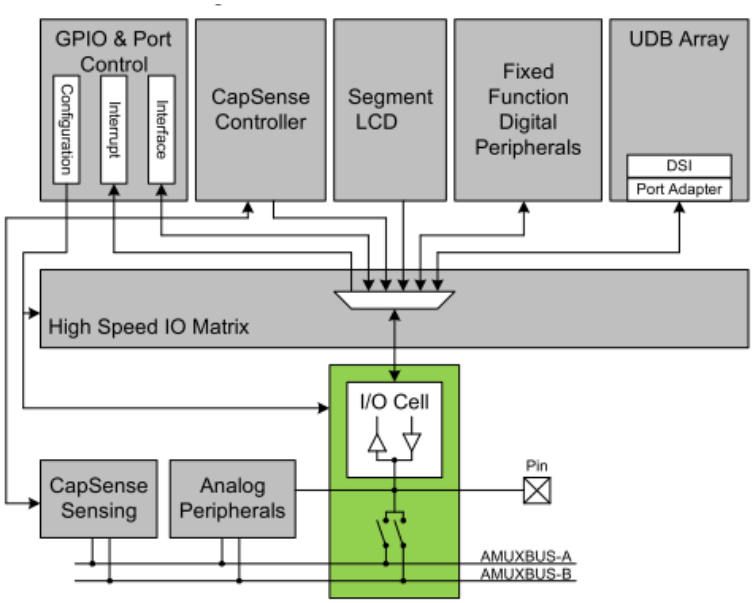


Рис. 3.1. Архітектура системи вводу/виводу мікроконтролера PSoC 6

На рис. 3.2 зображена архітектура комірки вводу/виводу, які є в кожній комірці GPIO. До складу комірки вводу/виводу входять вхідний буфер та вихідний драйвер, які під'єднуються до мультиплексорів HSIOM для цифрових вхідних та вихідних сигналів. Аналогові периферійні пристрої під'єднуються безпосередньо до виводу для з'єднання від точки до точки або з використанням AMUXBUS.

Буфер цифрового вводу забезпечує високоімпедансний буфер для зовнішнього цифрового вводу. Буфер включається або виключається бітом IN_EN[7: 0] регістру конфігурації порту (GPIO_PRTx_CFG, де x - номер порту). Вхідний буфер під'єднаний до HSIOM для маршрутизації на регістри порту ЦП та вибрані периферійні пристрої. Запис в регістр вибору порту HSIOM (HSIOM_PORT_SELx) вибирає вивід з'єднання.

Якщо вивід під'єднаний тільки до аналогового сигналу, вхідний буфер повинен бути від'єднаним, щоб уникнути струмів короткого замикання (crowbar currents). Точка спрацювання вхідного буферу кожного виводу та гістерезис налаштовуються для наступних режимів:

- CMOS + I^2C;
- TTL.

Ці режими буферу вибираються бітом VTRIP_SEL [7: 0]_0 регістру конфігурації вхідного буферу порту (GPIO_PRTx_CFG_IN).

Драйвер цифрового виводу. Виводи управляються драйвером цифрового виводу. Він складається зі схеми для реалізації різних режимів драйвера і управління швидкістю наростання для цифрових вихідних сигналів. Матриця HSIOM вибирає джерело управління для драйвера виводу. Існує три основних типи джерел управління - це регістри ЦП, цифрові периферійні пристрої, що конфігуруються, створені в програмованій вибірці UDB/DSI, та цифрові периферійні пристрої з фіксованою функцією. Конкретне з'єднання матриці HSIOM вибирається через запис в регістр вибору порту HSIOM (HSIOM_PORT_SELx).

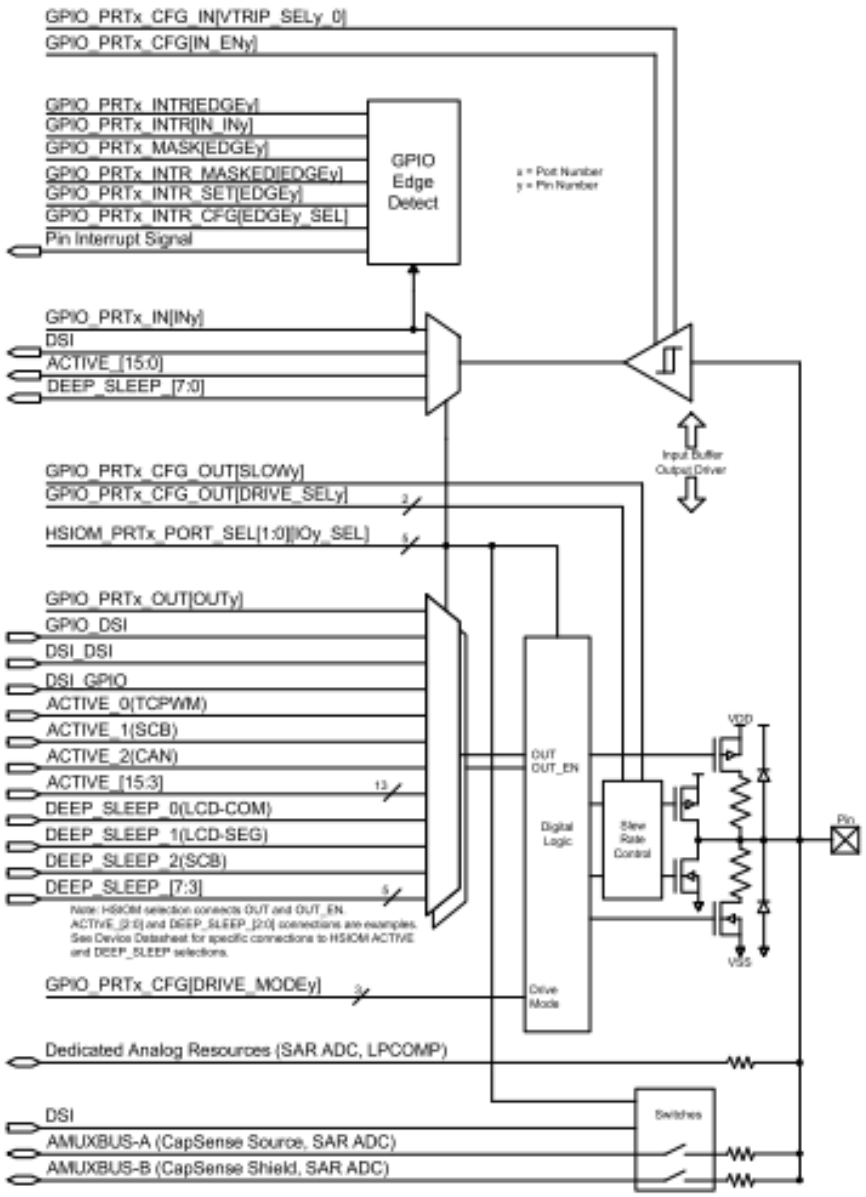


Рис. 3.2. Архітектура вводу/виводу комірки GPIO та GPIO_OVT

Шість груп входів/виходів живляться від різних джерел. В табл. 3.1. представлено розподіл портів в різних банках і джерела живлення відповідного банку.

Табл. 3.1. Банки вводу/виводу

Порти	Джерела живлення I/O
Port 0	V_{BACKUP}/V_{SSIO_B}
Port 1	V_{DDD}/V_{SSIO_B}
Port 2, Port 3, Port 4	V_{DDIO_R}/V_{SSIO_R}
Port 5, Port 6, Port 7, Port 8	V_{DDIO_1}/V_{SSIO_1}
Port 9, Port 10	V_{DDIO_A}/V_{SSIO_A}
Port 11, Port 12, Port 13	V_{DDIO_0}/V_{SSIO_0}

Кожний вивід GPIO має ESD діоди для фіксації напруги на виводі джерела живлення вводу/виводу. Необхідно переконатися, що напруга на виводі не перевищує напругу живлення вводу/виводу $V_{DDIO}/V_{DDD}/V_{DDA}$ або не падає нижче $V_{SSIO}/V_{SSD}/V_{SSA}$. Абсолютні максимальні і мінімальні напруги GPIO можна знайти в технічному описі мікроконтролера PSoC 6.

Драйвер цифрового виводу можна включить або відключити апаратно з допомогою сигналу DSI від периферійного пристрою або регістру вихідних даних (GPIO_PRTx_OUT), зв’язаного з вихідним виводом.

Стан вводу/виводу при включенні живлення. Під час включення живлення всі виводи GPIO знаходяться в високоімпедансному стані, вхідні буфери від’єднані. Під час виконання програми виводи GPIO можуть бути налаштовані, використовуючи відповідні регістри. Виводи, які підтримують з'єднання порту доступу (DAP) для налагодження (SWD-лінії), завжди включаються як SWD-лінії під час включення живлення. DAP-з'єднання не забезпечує підтягуючі ′вверх′ або ′вниз′ резистори. Тому, якщо залишити їх в плаваючому стані, то можливий струм короткого замикання (crowbar). З'єднання DAP можна відключити або налаштувати для загального використання через матрицю HSIOM лише після завантаження пристрою та початку виконання програми.

Регістр GPIO_PRTx_OUT використовується для читання і запису стану вихідного буфера для GPIO. Операція запису в цей регістр змінює стан вихідного драйвера GPIO на записане значення. Операція читання відображає вихідні дані, записані в цей регістр, і результуючий стан вихідного драйвера. Цей регістр не повертає поточний рівень логіки, який присутній на виводах GPIO, який може змінюватися. З допомогою регістру GPIO_PRTx_OUT послідовності читання - зміни – запису можна безпечно виконувати для порту, який має вхідний і вихідний GPIO.

Крім регістру даних передбачені ще три інших регістри - GPIO_PRTx_SET, GPIO_PRTx_CLR та GPIO_PRTx_INV – для встановлення, скидання і інвертування вихідних даних відповідно для певних виводів порту не змінюючи інші виводи. Це не вимагає необхідності в операціях читання-зміни-запису у більшості випадках використання. Запис логічної ′1′ у ці бітові поля регістру, встановить, скине або інвертує відповідний вивід, запису логічного ′0′ не вплине на стан виводу.

Аналогові ресурси, такі як LPCOMP, SAR ADC і CTB, які вимагають кіл з’єднання з низьким імпедансом, мають виділені виводи. Ці виділені аналогові виводи забезпечують пряме з’єднання з конкретними аналоговими блоками. Вони допомагають покращити продуктивність і повинні мати пріоритет над іншими виводами при використанні цих аналогових ресурсів.

Для конфігурації GPIO як виділений аналоговий ввід/вивід, він повинен бути сконфігурований в аналоговому режимі з високим імпедансом з від’єднаним вхідним буфером. Відповідне з’єднання повинно бути дозволене через регістри в конкретному аналоговому ресурсі.

Щоб налаштувати GPIO як аналоговий вивід, що під’єднується до AMUXBUS, його необхідно налаштувати в аналоговому режимі з високим імпедансом, від’єднавши вхідний буфер, а потім перенаправити його на AMUXBUS за допомогою регістра HSIOM_PORT_SELx.

Хоча для аналогових виводів бажано від’єднати вхідний буфер, допустимо включати вхідний буфер, якщо потрібні одночасні функції аналогового та цифрового вводу.

Існує два способи під’єднання виводу до AMUXBUS A або B. Вивід може бути статично зв’язаний з записом в регістр або мати під’єднання, динамічно

кероване з допомогою логіки на основі UDB, що визначається через DSI. Статичне з'єднання встановлюється з допомогою вибору AMUXA або AMUXB в регістрі HSIOM_PORT_SELx. Динамічне апаратно-кероване з'єднання виконується вибором AMUXA_DSI або AMUXB_DSI в регістрі HSIOM_PORT_SELx, що дозволяє реалізувати апаратне під'єднання AMUXBUS. Для налаштування виводу як вхід AMUXBUS потрібно виконати наступні дії:

1. Якщо з'єднання динамічно контролюється апаратними засобами, вихідні сигнали виводу і дозволу повинні бути під'єднані до сигналу вибору виводу, який генерується і направляється на вивід системою UDB/DSI.
2. Встановити вивід в режим високого імпедансу з від'єднаним вхідним буфером, що забезпечує аналогове під'єднання до виводу. Це конфігурування виконується з допомогою регістру GPIO_PRTx_CFG.
3. Сконфігурувати регістр HSIOM_PRT_SELx для під'єднання виводу до AMUXBUS A або B. Для статичних під'єднань потрібно вибрати AMUXA або AMUXB. Для динамічних під'єднань – вибрати AMUXA_DSI або AMUXB_DSI.

*LCD прив*ід. Всі GPIO мають можливість керувати LCD індикатором. Регістри HSIOM_PORT_SELx використовуються для вибору виводів для драйвера LCD.

Сенсор дотику CapSense. Виводи, які підтримують CapSense, можуть бути налаштовані як віджети CapSense, такі як кнопки, елементи повзунка, елементи сенсорної панелі або давачі наближення.

Інтелектуальні вводи/виводи. Інтелектуальний блок вводу/виводу додає програмовану логіку до порту вводу/виводу. Ця програмована логіка інтегрує в порт логічні функції на рівні стенду, такі як AND, OR та XOR. Блок Smart I/O має такі особливості:

- інтеграція функціональності булевої логіки на рівні плати в порт;
- можливість попередньої обробки вхідних сигналів HSIOM з виводів порту GPIO;
- можливість пост-обробки вихідних сигналів HSIOM на виводи порту GPIO.

Синхронізація вводу/виводу. Для цифрових вхідних і вихідних сигналів, пов'язаних з масивом Universal Digital Blocks (UDB), ввід/вивід забезпечує синхронізацію з внутрішніми тактовими імпульсами або цифровим сигналом, який використовується як тактовий сигнал. За замовчуванням HFCLK використовується для синхронізації, але також можна використовувати будь-який інше джерело тактових імпульсів. Ця функція реалізована за допомогою адаптера портів UDB.

Переривання портів вводу/виводу.

Всі виводи порту мають можливість генерувати переривання. Існує дві можливості маршрутизації сигналів виводів для генерації переривань (рис. 3.3):

- подати сигнал виводу через блок "GPIO Edge Detect" з прямим під'єднанням до контролера переривань процесора;
- подати сигнал через адаптер порту і Digital Signal Interface (DSI) на контролер переривань процесора.

На рис. 3.4 зображена архітектура блоку GPIO Edge Detect.

Детектор фронтів присутній на кожному виводі. Він здатний виявити передній (додатний) фронт, задній (від'ємний) фронт, обидва фронти без жодної ре конфігурації. Детектор фронтів налаштовується через записи в біти EDGEv_SEL регістру конфігурації переривань порту GPIO_PRTx_INTR_CFG, як зображено в табл. 3.2.

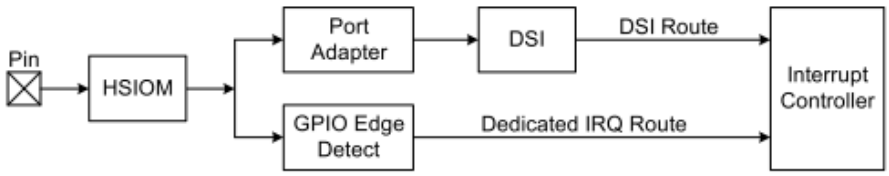


Рис. 3.3. Маршрутизація сигналу переривання

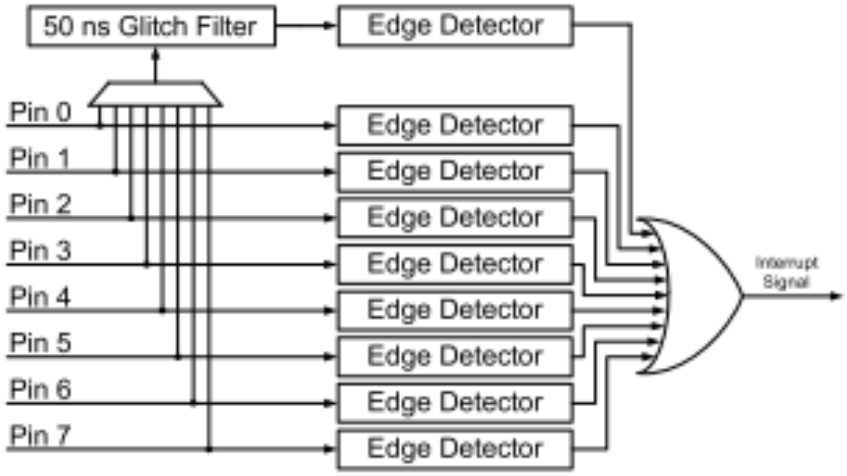


Рис. 3.4. Архітектура блоку GPIO Edge Detect

Запис логічної ′1′ у відповідний біт стану очищує стан фронту виводу. Це є важливим моментом при роботі з перериваннями, так як в іншому випадку переривання може виникати повторно для одного тригера або відповідати тільки один раз для декількох тригерів. Коли регістр статусу контролю переривань порту зчитується в той же час, коли на відповідному порту виникає фронт, це може призвести до того, що фронт не буде належним чином виявленим. Тому при використанні переривання GPIO потрібно зчитувати регістр стану тільки всередині відповідної підпрограми обробки переривань, а не в іншій частині програми.

Табл. 3.2. Конфігурація детектора фронту

EDGE_SEL	**Конфігурація**
00	Переривання заборонені
01	Переривання по передньому фронті
10	Переривання по задньому фронті
11	Переривання по обох фронтах

Вбудоване мікропрограмне забезпечення і інтерфейс налагодження можуть запускати апаратне переривання з будь-якого виводу, встановлюючи відповідний біт в регістрі GPIO_PRTx_INTR_SET. Кожний порт крім виводів має фільтр завад, під'єднаний до свого детектора фронту. Цей фільтр може керуватися одним з виводів порту. Вибір ведучого виводу виконується через запис в поле FLT_SEL регістру GPIO_PRTx_INTR_CFG (табл. 3.3).

Якщо виникає фронту виводу порту, то можна прочитати регістр стану переривання порту GPIO_PRTx_INTR, щоб дізнатися який вивід викликав

появу фронту. Цей регістр включає в себе як фіксовану інформацію про те, який вивід виявив фронт, так і поточний статус виводу. Це дозволяє ЦП прочитати обидві інформації за одну операцію читання. Цей регістр також використовується для очистки стану заблокованого фронту.

Табл. 3.3. Конфігурація детектора фронту

FLT_SEL	Вибір виводу
000	Вибрано вивід 0
001	Вибрано вивід 1
010	Вибрано вивід 2
011	Вибрано вивід 3
100	Вибрано вивід 4
101	Вибрано вивід 5
110	Вибрано вивід 6
111	Вибрано вивід 7

Регістр GPIO_PRTx_INTR_MASK дозволяє пересилати сигнал виявлення фронту GPIO_PRTx_INTR до контролера переривання, коли логічна ′1′ записується у відповідне бітове поле виводу. Потім можна прочитати регістр GPIO_PRTx_INTR_MASKED, щоб визначити конкретний вивід, який генерував сигнал переривання, що передається на контролер переривання. Потім виводи маскованого детектора фронту порту об’єднуються і направляються на контролер переривання (NVIC в підсистемі ЦП). Таким чином, існує лише один вектор переривання на один порт.

Вихід блоку детектора маскованих і Ored фронтів передається на мультиплексор джерела переривань, який має можливість виявлення рівня і наростання фронту. Якщо вибрана опція Рівень (Level), переривання запускається багатократно, поки встановлений біт регістру стану переривання порту. Якщо ж вибрана опція виявлення переднього фронту (Rising Edge), переривання спрацьовує тільки один раз, якщо регістр стану переривання не очищений. При використанні блоку детектування фронту (Edge Detect) біт стану переривання повинен бути очищений. Кожний порт має виділений вектор переривання, якщо сигнал переривання направляється по маршруту з фіксованою функцією. Якщо ж сигнал направляється через DSI, вектор переривання є гнучким і може займати будь-яку з ліній переривання, пов’язаних з DSI, NVIC.

Всі вектори переривань портів також об’єднуються в один вектор переривань для використання на пристроях з більшою кількістю портів, ніж існує в наявності векторів переривань. Для виявлення порту, що викликав переривання, можна прочитати регістри GPIO_INTR_CAUSEx. Наявність логічної ′1′ в певному біті вказує, що відповідний порт має очікуване переривання. Потім цей регістр GPIO_PRTx_INTR може бути зчитаний для визначення джерела виводу.

У випадку, коли сигнал направляється через DSI, минаючи блок Edge Detect, детектування фронту налаштовується в блоці мультиплексора джерела переривання. Якщо мультиплексор налаштовано по рівню, переривання запускається багатократно, поки сигнал виводу в стані логічної ′1′. Тому

потрібно використовувати опцію виявлення переднього фронту, якщо вибрано цей маршрут, щоб згенерувати тільки одне переривання.

Загальний опис компоненту GPIO.

Компонент GPIO - це графічний об'єкт конфігурації, побудований поверх драйвера cy_gpio, доступного в бібліотеці периферійних драйверів (PDL - Peripheral Driver Library). Він дозволяє з'єднання на основі схеми та апаратну конфігурацію, як це визначено в діалоговому вікні ″Налаштування компонентів″ (Component Configure).

Компонент GPIO дозволяє апаратним ресурсам під єднатися до фізичного виводу порту. Він забезпечує доступ до зовнішніх сигналів через правильно налаштований фізичний вивід вводу/виводу. Також він дозволяє вибирати електричні характеристики (наприклад, режим драйвера) для одного або декількох виводів. Ці характеристики потім використовуються середовищем розробки PSoC Creator для автоматичного розміщення та маршрутизації сигналів в компоненті.

Компонент GPIO можна використовувати із з'єднанням дротів зі схемою, програмним забезпеченням або обома. Він може бути налаштований на багато комбінацій типів. Для зручності Каталог компонентів містить чотири попередньо налаштовані компоненти GPIO: аналоговий, цифровий двонаправлений, цифровий вхід та цифровий вихід.

Компонент GPIO використовується, коли в проекті необхідно генерувати або отримувати доступ до сигналу поза мікроконтролером через фізичний вивід вводу/виводу. Це зручний спосіб налаштовувати виводи в графічному вигляді і дозволяє PSoC Creator IDE виконувати маршрутизацію та мультиплексування виводів.

Відображення GPIOs можна налаштувати на складні комбінації цифрового входу, цифрового виходу, цифрового двонаправленого та аналогового виводів. Прості конфігурації, як правило, відображаються у вигляді окремих виводів. Більш складні типи виводів показані як стандартні компоненти з обмежувальною рамкою.

Параметри компонентів.

Для того, щоб відкрити діалогове вікно Налаштування (Configure) потрібно перетягнути компонент GPIO на схему проекту та двічі клацнути на ньому лівою кнопкою мишки (рис. 3.5). Це вікно використовується для встановлення параметрів компонент, які організовані в окремі вкладки.

Вкладка Виводи (Pins) має три області: панель інструментів, дерево виводів і набір вкладених вкладок. Панель інструментів використовується для визначення скількома фізичними виводами керує компонент і визначення їх порядку. Вкладені вкладки використовуються для задання специфічних для виводів атрибутів, таких як тип виводів, режим драйвера і початковий стан драйвера Дерево виводів працює з вкладеними вкладками для вибору конкретних виводів, для яких ці атрибути використовуються.

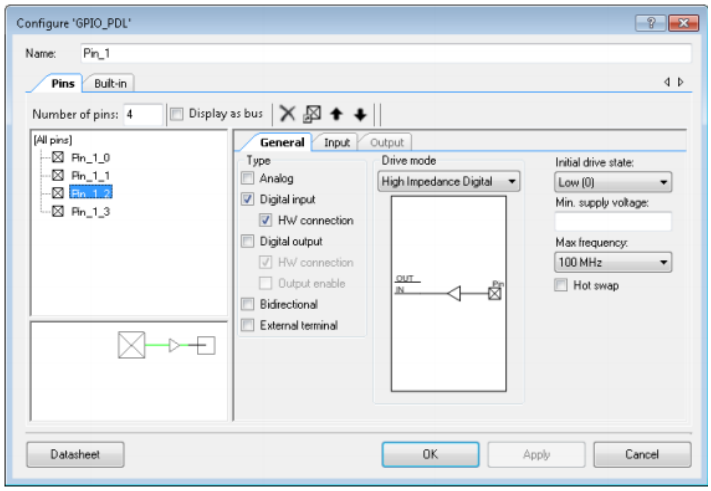


Рис. 3.5. Вигляд вікна налаштування виводів GPIO PDU

Панель інструментів містить наступні команди:

- Число виводів. (Кількість виводів пристрою, які контролюються Компонентом. Значення за замовчуванням - 1.)

- Відображення шини.

Цей параметр вибирає, чи відображати окремі термінали для кожного виводу або один широкий термінал (шину). Варіант шини дійсний лише тоді, коли виводи однорідні. Це означає, що всі виводи компонента мають однаковий тип, вхідні/вихідні з'єднання HW та групування SIO. Всі вони також повинні використовувати або не використовувати SIO Vref.

- Видалення виводу. (Видаляє вибрані виводи з дерева)

- Додати/змінити ім’я. (Відкриває діалогове вікно для додавання або зміни імені для вибраного виводу в дереві. Також можна відкрити діалогове вікно двічі клацнувши на виводі або натиснувши [F2])

- Перемістити вверх/вниз. (Переміщує вибрані виводи вверх або вниз по дереву)

- З’єднувати/не з’єднувати. (З'єднує або скасовує вибрані SIO виводи позначені рожевим контуром у дереві)

Цей елемент керування визначає, чи потрібно виводи, які потребують (SIO - special I/O), розміщувати в одній і тій же парі SIO на пристрої. Узгодження виводів призводить до того, що менше фізичних SIO виводів "витрачається". Щоб виводи мали спільну пару SIO на пристрої, вони повинні мати однакові налаштування для кожної пари і бути суміжними.

Для виводу потрібно SIO, якщо вибрано функцію ″Гаряча заміна″ (Hot Swap), ждя параметру ″Поріг″ (Threshold) задано будь-яке значення крім LVTTL або CMOS, для рівня драйвера встановлено значення "Vref" і/або для параметру ″Струм″ встановлено значення ″25 мА споживання (sink)″.

В області ″Дерево виводів″(Pin Tree) відображаються всі виводи для компоненту. Можна індивідуально вибрати один або декілька виводів для використання з командами панелі інструментів і вкладеними вкладками. Кожний вивід відображає своє ім’я, яке складається з <Імені екземпляра GPIO>_<Індивідуального імені виводу>.

Під деревом знаходиться область попереднього перегляду, яка показує який вигляд матиме вибраний символ компоненту GPIO з різними параметрами, вибраними для цього конкретного виводу.

Загальна підвкладка. Ця підвкладка за замовчуванням, що відображається на вкладці "Виводи" зображена на рис 3.6.

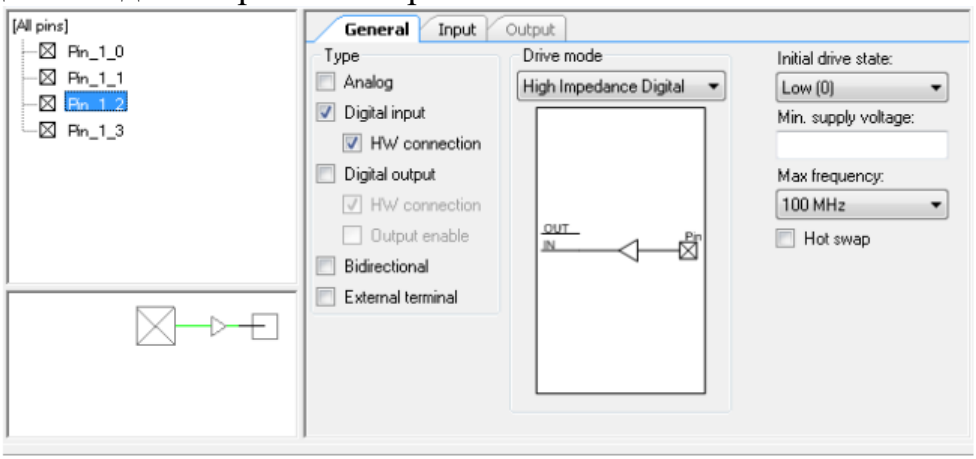


Рис. 3.6. Вигляд загальної підвкладки вкладки "Виводи"

Вона містить такі параметри:

На ній вибираємо тип виводів для свого компонента за допомогою прапорців.

- Аналоговий. (Вибираємо "Аналоговий", щоб увімкнути термінал аналогового виводу. Це дозволяє маршрутизацію аналогового сигналу до інших компонентів. Вибір аналогового виводу змушує вивід фізично розміщуватися на виводі GPIO, а не на виводі SIO).

- Цифровий вхід. (Вибираємо "Цифровий вхід", щоб увімкнути термінал цифрового входу для вводу (необов'язково) та включити вкладку "Вхід" для додаткових параметрів конфігурації, пов'язаних з входами.)

- HW з'єднання. (Цей параметр визначає чи буде цифровий вхідний термінал для вхідного виводу відображатися на схемі. Якщо відображається, то вивід забезпечує цифровий сигнал, який може під'єднуватися до інших цифрових приймачів, таких як термінали компонентів. Якщо ця опція не вибрана, термінал не відображається і може бути прочитаний тільки CPU/DMA).

- Цифровий вихід. (Вибираємо "Цифровий вихід", щоб увімкнути термінал цифрового виходу (необов'язково) та включити підвкладку "Вихід" для додаткових параметрів конфігурації, пов'язаних з виходами.)

- HW з'єднання. (Цей параметр визначає, чи буде цифровий вихідний термінал для заданого вихідного виводу відображається на схемі. Якщо відображається, вивід виводить цифровий сигнал, що подається апаратним джерелом. Якщо не відображається, рівень вихідної логіки визначається записом в ЦП. Якщо ця опція не вибрана, термінал не відображається, а ним керує лише CPU/DMA. Коли вибрано цей параметр, початковий стан драйверу виводу встановлюється в стан логічної '1'. Це параметр неможливо змінити. Це налаштування може призвести до збою на виводі під час ініціалізації пристрою. Щоб уникнути такої поведінки, спеціалісти фірми Cypress рекомендують налаштувати вивід як "Аналоговий високий z" в компоненті, а потім змінити режим драйвера виводу на "Сильний драйвер" у прошивці після ініціалізації та включення компонента, до якого він під'єднаний.)

- Output Enable (включення виходу). (Цей параметр дозволяє використовувати функцію включення виходу виводу і відображає вхідний термінал дозволу виводу. Функція дозволу виходу дозволяє апаратному (DSI)

сигналу керувати драйверами виходів виводу без ЦП. Рівень логічної ′1′ конфігурує драйвери виходу як встановлено в параметрі ″Режимів драйвера″ (Drive Mode). Низький логічний рівень від'єднує драйвери віходу і переводить вивід в режим високоомного драйвера.)

Двонаправлений вивід. (Дозвіл двонаправленого параметра функціонально еквівалентний включенню цифрового входу з під'єднанням HW і цифрового виходу з параметром під'єднання HW. Різниця полягає в тому, що на символі компоненту відображається тільки один двонаправлений термінал, а не окремі вхідні та вихідні виводи. Потім термінал може під'єднуватися до інших з'єднань двонаправленого типу. Як вхідна, так і вихідна вкладені вкладки включені для подальшого налаштування.)

- Показати зовнішній термінал. (Дозволяє з'єднання з зовнішніми компонентами в Каталозі компонентів для ілюстрації електричної схеми зовнішнього чипу,)

Режим драйвера. Цей параметр налаштовує вивід для забезпечення одного з 8- ми доступних режимів драйвера виводу. Значення за замовчуванням і логічний вибір залежать від вибору типу, Діаграма показує представлення схеми для кожного вибраного режиму драйвера.

- Якщо тип є цифровим входом або цифровим входом/аналоговим за замовчуванням використовується високоімпедансний цифровий вхід.
- Якщо тип є аналоговим, за замовчуванням використовується аналоговий високоімпедансний вхід. Це єдиний режим драйвера виводу, який може підтримувати тільки аналогові виводи.
- Якщо тип є двонаправленим виводом або двонаправленим/аналоговим, за замовчуванням встановлено ″Відкритий стік″, ″Низький рівень драйвера″.
- Всі інші типи виводу за замовчуванням встановлені в ″Сильний драйвер″.

Ініціалізація стану драйвера. Цей параметр вказує специфічне для виводу початкове значення, записане в регістр OUT виводів після скидання/включення пристрою. Це відбувається під час процесу налаштування порту в коді запуску пристрою. Якщо не змінювати вручну або автоматично налаштувати в стан логічної ′1′ з допомогою параметру ″Режим драйвера″, всі виводи за замовчуванням мають рівень логічного ′0′. Початковий стан драйвера за замовчуванням налаштований на рівень логічної ′1′ тільки для ″Резистору″.

Конфігурація виводу.

Інструменти конфігурації пристрою, такі як PSoC Creator, автоматично генерують код конфігурації GPIO та виконують його як частину процесу завантаження пристрою в cyfitter_cfg.c. Методи конфігурації GPIO, як правило, використовуються лише з ручним налаштуванням PDL GPIO, коли не використовується інструмент конфігурації. Вони також можуть використовуватися під час роботи для динамічного переналаштування виводів GPIO незалежно від того, як виконувалася початкова конфігурація.

Більшість виводів GPIO вимагають встановлення лише своїх основних параметрів і можуть використовувати значення за замовчуванням для всіх інших параметрів. Це дозволяє використовувати спрощену функцію ініціалізації. Функція Cy_GPIO_Pin_FastInit () підтримує лише параметризовану конфігурацію режиму пристрою, рівня логічного виходу та налаштування швидкісного мультиплексора вводу/виводу (HSIOM). Налаштування HSIOM

визначає програмне забезпечення високого рівня, периферійне та аналогове управління та підключення. Усі інші налаштування конфігурації не змінюються після їх скидання або раніше встановленого стану. Ця функція дуже корисна під час виконання, щоб динамічно змінювати конфігурацію виводу. Наприклад, налаштуємо вивід в режим сильного драйвера для запису даних, а потім переналаштуємо вивід в режим високого імпедансу для зчитування даних.

```
Cy_GPIO_Pin_FastInit(P0_3_PORT, P0_3_NUM, CY_GPIO_DM_STRONG, 1, HSIOM_SEL_GPIO);
```

Метод налаштування всіх атрибутів одного виводу полягає у використанні функції Cy_GPIO_Pin_Init () та структури конфігурації виводів. Хоча він простий у використанні, він генерує більший код, ніж інші методи конфігурації.

```
Cy_GPIO_Pin_Init(P0_4_PORT,P0_4_NUM, &P0_4_Pin_Init);
```

Найбільш ефективним методом для налаштування всіх атрибутів для повного порту виводів є використання функції Cy_GPIO_Port_Init () та структури конфігурації порту. Він упакує всі дані конфігурації в прямі записи в регістр для всього порту. Його обмеження полягає в тому, що він повинен налаштувати всі виводи в порту, і користувач повинен обчислити об'єднані значення регістрів для всіх виводів або скопіювати їх з інструмента конфігурації. Цей метод, застосовується інструментами автоматичної конфігурації як частина процесу завантаження пристрою в cyfitter_cfg.c.

```
Cy_GPIO_Port_Init(GPIO_PRT5, &port5_Init);
```

Індивідуальні налаштування конфігурації виводів також можуть бути змінені під час виконання, використовуючи функції драйвера. Приклад деяких із цих функцій наведено нижче. Параметри функції демонструють використання специфічних для виводів #defines, представлених у cyfitter_gpio.h, коли використовується інструмент налаштування.

```
Cy_GPIO_SetHSIOM(SW2_P0_4_PORT, SW2_P0_4_NUM, SW2_P0_4_INIT_MUXSEL);
Cy_GPIO_SetDrivemode(SW2_P0_4_PORT, SW2_P0_4_NUM, CY_GPIO_DM_PULLUP);
Cy_GPIO_SetVtrip(SW2_P0_4_PORT, SW2_P0_4_NUM, SW2_P0_4_THRESHOLD_LEVEL);
Cy_GPIO_SetSlewRate(SW2_P0_4_PORT, SW2_P0_4_NUM, SW2_P0_4_SLEWRATE);
Cy_GPIO_SetDriveSel(SW2_P0_4_PORT, SW2_P0_4_NUM, CY_GPIO_DRIVE_FULL);
```

Методи читання даних з виводу.

Існують декілька методів зчитування даних з виводу GPIO. Тому для кожного конкретного випадку використання потрібно вибрати найбільш відповідний метод. Функція Cy_GPIO_Read () безпечна для потоків та багатоядерних процесорів. Більшість функцій драйверів GPIO потребують як мінімум двох аргументів для визначення порту та виводу в цьому порту, на якому він працює. Аргумент Port очікує базової адреси регістрів порту. Аргумент Pin очікує на номер вивода цього порту. Кращим способом зчитування для використання з інструментами конфігурації є використання #defines, передбачених для імен виводів інструмента конфігурації. У PSoC Creator - це компоненти виводу, розміщені на схемі, а вивід #defines розміщений у \\pdl\Pins та Interrupts\cyfitter_gpio.h.

```
pinReadValue = Cy_GPIO_Read(SW2_P0_4_PORT, SW2_P0_4_NUM);
```

Кращим методом зчитування для прямого використання PDL без інструменту конфігурації є використання користувацьких імен #define pin.

Користувацькі #defines зазвичай розміщуються у створеному користувачем *.h файлі.

```
#define mySwPin_Port P0_4_PORT
#define mySwPin_Num P0_4_NUM
pinReadValue = Cy_GPIO_Read(mySwPin_Port, mySwPin_Num);
```

Також можна прочитати виводи, використовуючи ім'я виводу за замовчуванням для пристрою #defines, яке є передбаченим для кожного пристрою та пакету. Ім'я за замовчуванням #defines знаходиться в _mainapp_(device series)pdl\Includes\(.. devices/series/include)\gpio_(series+ package) .h.

```
pinReadValue = Cy_GPIO_Read(P0_4_PORT, P0_4_NUM);
```

Читання виводу з використанням номеру порту та номера виводу порту також підтримується. Цей метод корисний для автоматично генерованих портів і виводів. Cy_GPIO_PortToAddr() – це допоміжна функція, яка перетворює номер порту в потрібну базову адресу регістру порту, яка потрібна для інших функцій драйвера GPIO.

```
portNumber = 0;
pinReadValue = Cy_GPIO_Read(Cy_GPIO_PortToAddr(portNumber), 4);
```

Як і для будь-якого мікроконтролера, прямий доступ до регістру портів завжди доступний і корисний для одночасного доступу до декількох виводів порту або для розробки оптимізованого для додатків доступу до портів. В наступному прикладі показано регістр порту IN, зчитаний з маскою і зміщенням потрібних даних виводу.

```
pinReadValue = (GPIO_PRT0->IN >> P0_4_NUM) & CY_GPIO_IN_MASK;
```

Методи запису даних в вивід виходу.

Наведені нижче методи виконують один і той ж запис в виводи GPIO, використовуючи різні доступні методи запису. Завжди вибирається найбільш відповідний метод для конкретного випадку використання. Функцію Cy_GPIO_Write () найкраще використовувати, коли потрібний стан виводу ще не відомий і визначається під час виконання. Функція запису використовує атомні операції, які безпосередньо впливають лише на вибраний вивід без використання операцій читання-зміни-запису. Таким чином, функція запису є безпечною для потокових та багатоядерних.

Переважним методом прямого використання PDL без інструменту конфігурації є запис виводів із користувацькими іменами #define. Надані користувачем #defines зазвичай розміщуються у створеному користувачем .h файлі.

```
#define   myLedPin_Port P1_5_PORT
#define   myLedPin_Num P1_5_NUM
Cy_GPIO_Write(myLedPin_Port, myLedPin_Num, pinReadValue);
```

Також можна записати дані в виводи, використовуючи ім'я виводу за замовчуванням для пристрою #defines, розміщеного в _mainapp_(device series)pdl\Includes\(..devices/series/include)\gpio_(series + package).h file

```
Cy_GPIO_Write(P1_5_PORT, P1_5_NUM, pinReadValue);
```

Запис в виводи можлива з використанням імені порту за замовчуванням #defines і номера виводів. #defines знаходиться в файлі _mainapp_(device series)pdl\Includes\(..devices/series/include)\(part number) .h .

```
Cy_GPIO_Write(GPIO_PRT1, 5, pinReadValue);
```

Для алгоритмічно генерованих номерів портів і виводів корисними є записи в виводи з використанням номерів портів і виводів. Функція Cy_GPIO_PortToAddr () перетворює номер порту в потрібну базову адресу регістру порту.

```
portNumber = 1;
Cy_GPIO_Write(Cy_GPIO_PortToAddr(portNumber), 5, pinReadValue);
```

Найбільш ефективні методи виводу, коли бажаний стан виводів вже відомий під час компіляції, - це безпосередньо встановлювати, очищати та інвертувати стан виводів. Ці записи в регістр є атомарними операціями, які напряму впливають тільки на вибраний вивід без використання операцій читання-зміни-запису. Тому вони є безпечними як для багато потокових, так і для багатоядерних систем. Можуть також використовуватися ті ж варіанти аргументів, які продемонстровані з допомогою функції Cy_GPIO_Write ().

```
Cy_GPIO_Set(KIT_LED1_PORT, KIT_LED1_NUM);
Cy_GPIO_Clr(KIT_LED1_PORT, KIT_LED1_NUM);
Cy_GPIO_Inv(KIT_LED1_PORT, KIT_LED1_NUM);
```

Доступ до порту.

Прямий доступ до регістру використовується для взаємодії з декількома виводами в одному порту одночасно. Ці звертання не можуть бути багатоядерними через можливі операції читання-зміни-запису. Всі виводи в порту, які знаходяться під прямим контролем регістру, повинні бути доступними тільки одному ядру центрального процесора, якщо захист доступу не забезпечений на системному рівні.

```
portReadValue = GPIO_PRT7->IN;
portReadValue++;
GPIO_PRT7->OUT = portReadValue;
```

Переривання по виводах.

Для того, щоб згенерувати переривання по виводу, потрібно налаштувати його так, щоб воно спрацьовувало по додатному, від’ємному або обох фронтах сигналу і замаскувати його так, щоб сигнал виводу передавався на вектор контролера переривань для цього порту.

```
Cy_GPIO_SetInterruptEdge(KIT_BTN1_PORT, KIT_BTN1_NUM, CY_GPIO_INTR_RISING);
Cy_GPIO_SetInterruptMask(KIT_BTN1_PORT, KIT_BTN1_NUM, CY_GPIO_INTR_EN_MASK);
```

Вектор переривання порту повинен бути потім сконфігурований, очищений та дозволений для запуску з сигналу порту і співставлений з потрібною підпрограмою обробки переривання (Interrupt Service Routine - ISR). Для отримання додаткової інформації про конфігурування та використання переривань можна переглянути документацію про PDL Cy_SysInt.

```
Cy_SysInt_Init(&intrCfg, GPIO_Interrupt);
NVIC_ClearPendingIRQ(intrCfg.intrSrc);
NVIC_EnableIRQ((IRQn_Type)intrCfg.intrSrc);
```

Після того, як відбулося переривання, перед виходом з ISR, потрібно очистити переривання цього виводу, щоб логіка детектування фронту була скинута, щоб дозволити детектування наступного фронту сигналу.

```
void GPIO_Interrupt()
 {
  Cy_GPIO_ClearInterrupt(KIT_BTN1_PORT, KIT_BTN1_NUM);
```

```
}
```

Якщо може генеруватися переривання для декількох виводів порту, функція Cy_GPIO_GetInterruptStatus () може використовуватися для ідентифікації виводу, який детектував появу фронту сигналу та згенерував переривання. Для визначення виводу переривання може використовуватися пряме зчитування регістру INTR.

```
Cy_GPIO_ClearInterrupt(KIT_BTN1_PORT, KIT_BTN1_NUM);
portIntrStatus = KIT_BTN1_PORT->INTR;
if(CY_GPIO_INTR_STATUS_MASK == ((portIntrStatus >> KIT_BTN1_NUM) &
                                CY_GPIO_INTR_STATUS_MASK))
```

В багатьох проектах кожному порту, який може генерувати переривання, буде назначено свій власний вектор переривання. Якщо обмеження вектора переривання не дозволяють використовувати дискретні вектори, може використовуватися комбіноване переривання порту (AllPortInt). AllPortInt Ors об’єднує всі сигнали переривання окремих портів в один сигнал всього кристалу, який вимагає тільки одного вектора для довільної кількості виводів на декількох портах. Функція Cy_GPIO_GetInterruptCause0 () може використовуватися для ідентифікації порту(ів), які детектували подію фронт сигналу для переривання.

```
if((Cy_GPIO_GetInterruptCause0() & INTCAUSE0_PORT0) != 0)
```

Приклад проекту.

Реалізуємо проект, який демонструє декілька методів налаштування, читання, запису та генерування переривань за допомогою виводів вводу/виводу загального призначення (GPIO) мікроконтролера PSOC 6. Схема цього проекту зображена на рис. 3.7.

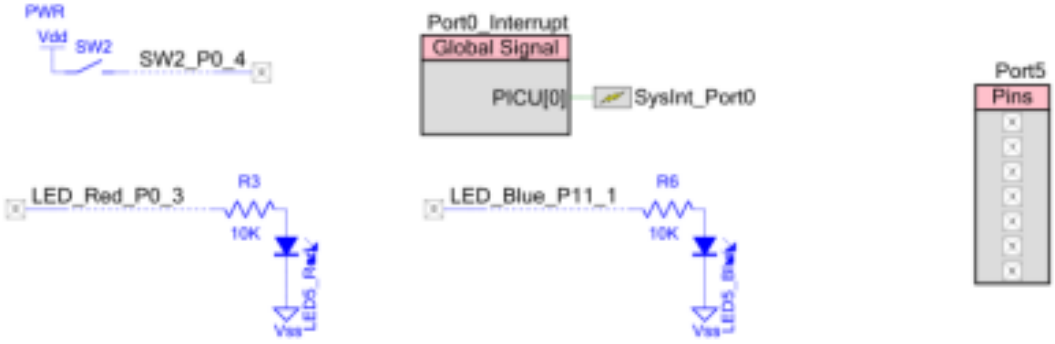


Рис. 3.7. Схема проекту для демонстрації методів налаштування, читання, запису, генерації переривань для виводів GPIO PSoC 6

Для демонстрації індивідуального доступу до окремих виводів GPIO у цьому проекті використовується цифровий вивід, налаштований на вхід. До нього під’єднано кнопку SW2 стенду. Стан кнопки постійно зчитується процесором Cortex M4. Зчитане значення стану кнопки записується в цифровий вивід, який під’єднаний до світлодіоду RGB (LED_Red) стенду. Після натискання кнопки горить червоний світлодіод.

Цифровий вхідний вивід також сконфігурований з можливістю генерувати переривання по задньому фронту, яке відбувається при відпусканні натиснутої

кнопки. Програма переривання засвічує RGB (LED_Blue) світлодіод на другому виводі цифрового вихідного виводу приблизно на 1 сек.

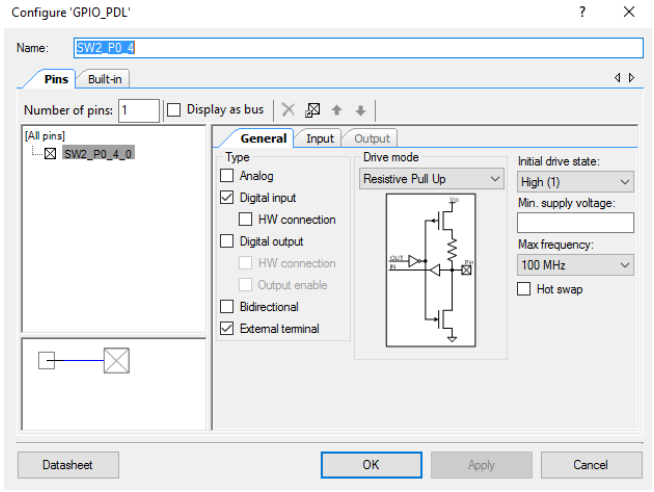
Повний порт виводів GPIO налаштовано за допомогою масиву компоненти виводів на Port 5. Значення Port 5 постійно зчитується за допомогою прямого читання регістру, збільшується вміст на одиницю та записується назад в порт. Стан виводів порту можна контролювати за допомогою осцилографа.

Завдання:

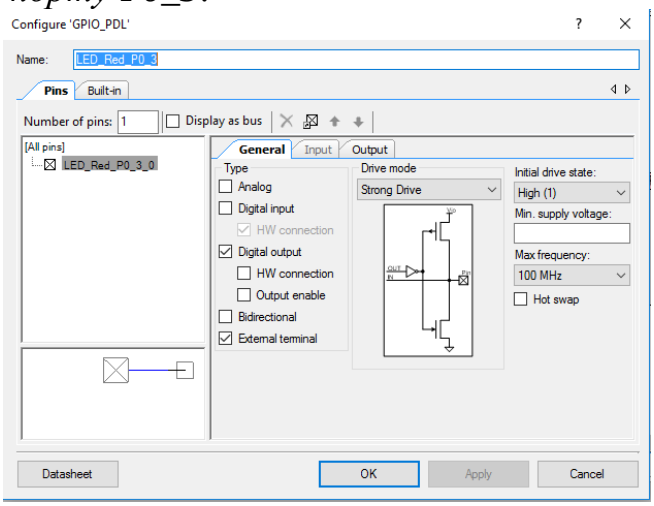
1. Реалізувати проект, принципова схема якого зображена на рис. 3.7. Програмну частину для нього взяти з файлу main_cm4.c проекту CE220263 [1].

Налаштування елементів цього проекту зображені на рисунках нижче.

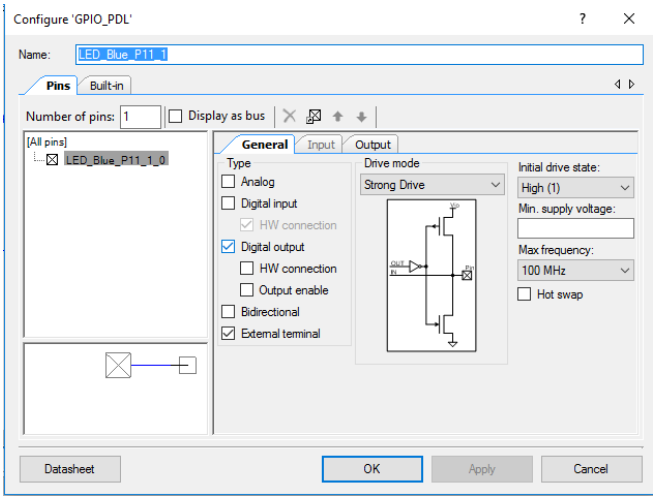
Налаштування порту P0_4:



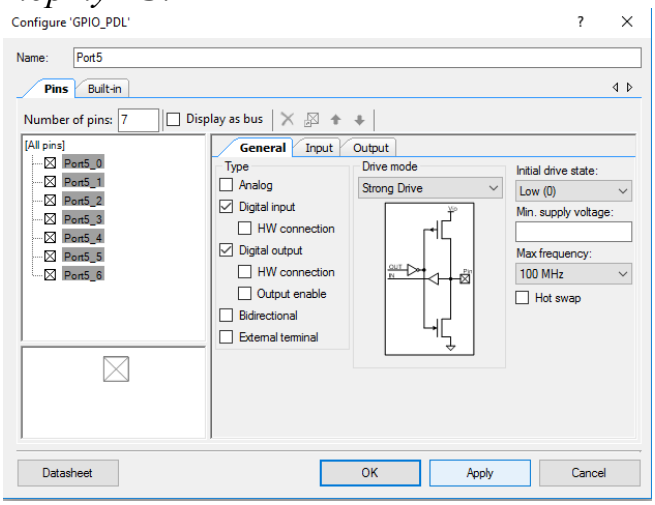
Налаштування порту P0_3:



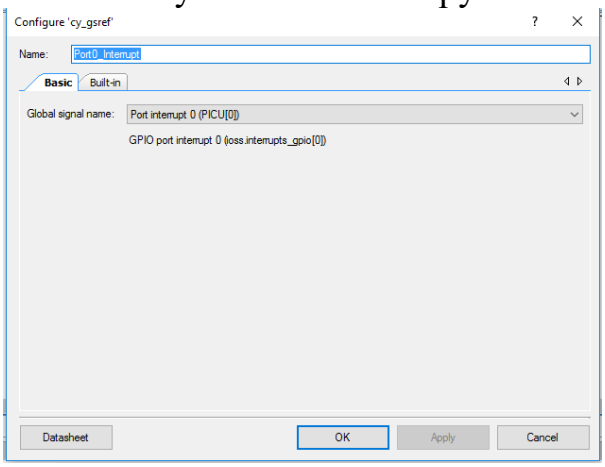
Налаштування порту P11_1:

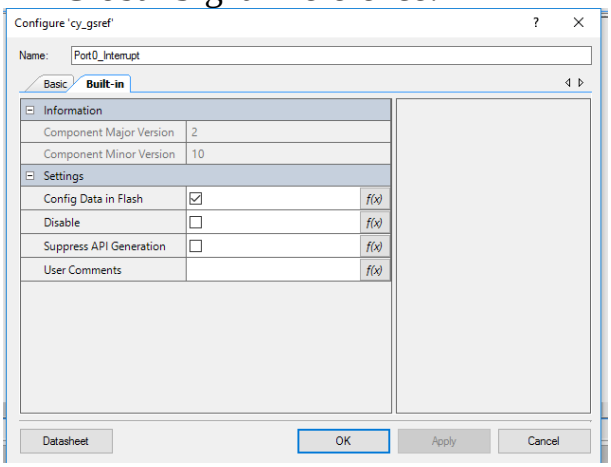


Налаштування порту P5:



Налаштування екземпляру компоненти Global Signal Reference:





2. Реалізувати проект, в якому засвічується зелений світлодіод RGB LED стенду з частотою 2 Гц, якщо кнопка SW2 не натиснута. При натисканні кнопки SW2 та її утриманні засвічується постійно червоний світлодіод RGB LED.

Література.

1. CE220263 – PSoC 6 MCU GPIO Pins Example. Електронний ресурс. – Режим доступу: https://www.cypress.com/file/385856/download