Лабораторна робота № 10

Вивчення роботи SAR ADC мікроконтролера PSoC 6

Мета роботи: Вивчити функціонування SAR ADC мікроконтролера PSOC 6.

Теоретичні відомості

Основні характеристики SAR ADC мікроконтролера PSoC 6.

- 12-розрядна роздільна здатність ADC;
- Почергове або канально - послідовне усереднення в обладнанні;
- До 16-ти розрядів роздільна здатність із усередненням;
- Загальна швидкість вибірки до 1 Msps;
- Односторонній та диференційний режими введення сигналу;
- Планувальник оптимізує встановлення часу та годинника відповідно до швидкості сканування;
- Автоматично сканувати до шістнадцяти аналогових сигналів;
- Чотири конфігурації, що вибираються під час роботи.

Скануючий SAR ADC - це компонент, який використовується для доступу до функцій АЦП у мікроконтролерах сімейства PSoC 6. Він є гнучким і універсальним як при неперервній вибірці з високою частотою дискретизації додатку (розраховані повністю на апаратне забезпечення), так і в спеціалізованих додатках сканування з більш низькою швидкістю.

Зміщення та діапазон АЦП залежать від параметрів налаштування компонента. Незалежно від цих налаштувань, аналогові сигнали, підключені до виводів PSoC, повинні знаходитися між V_{SSA} та V_{DDA}. Для деяких налаштувань можливе перетворення rail-to-rail (діапазон вхідних синфазних сигналів, що включає напруги живлення).

З'єднання входи/виходи.

Розглянемо різні підключення входів/виходів для АЦП послідовного наближення, які можуть відображатися у вигляді клем на символі компонента. Зірочка (*) після імені клеми вказує, що клема може бути відсутньою на символі при певних умовах. Використовуються скорочені записи підключення сигналів: "s/e" – односторонній сигнал; "diff" – диференційний сигнал. На протязі часу вибірки для даного каналу його вхідні сигнали +Input, -Input або vneg підключаються безпосередньо до вхідного конденсатора ядра АЦП і повинні зарядити цей конденсатор до фактичного перетворення. Мінімальне значення часу встановлення входу може бути введене в вибір параметрів кожного каналу, щоб врахувати повний опір джерела цього каналу.

Аналоговий вхід +Input.

Цей вхід (не відмічений) – це завжди верхня клема пари диференційних входів на символі. Він є "додатним" (неінвертуючим) аналоговим сигналом, що

поступає на АЦП. Кількість вхідних клем "додатного" аналогового сигналу завжди співпадає з кількістю вибраних каналів, не залежно від того, чи вони вибрані як диференційні або односторонні.

Символ, зображений на рис. 10.1, має два канали. Нульовий канал сконфігурований як односторонній канал з використанням vref як вхідного інвертуючого з'єднання.

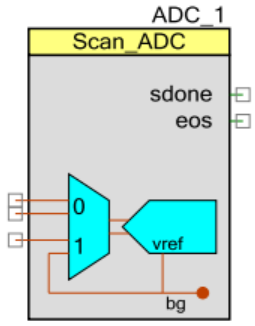

Рис. 10.1. Вигляд символу АЦП мікроконтролера PSoC 6

Аналоговий вхід -Input*.

Цей вхід (не відображається – це завжди нижня клема пари диференційних входів на символі) є "від'ємним" (інвертуючим) аналоговим сигналом, що поступає на АЦП. Він відображається тільки для каналів, які були оголошені як диференційні. На всіх каналах, оголошених як односторонні канали, інвертуючий вхід АЦП підключений до сигналу Vneg. Завжди є однакова кількість "від'ємних" вхідних клем аналогового сигналу, так як вибрані диференційні канали.

Аналоговий вхід - Vneg*.

Це загальний від'ємний вхідний еталон. Ця клема є присутня тільки тоді, коли один або декілька аналоговий каналів оголошені як односторонній вхід, а для параметра Vneg для S/E встановлено значення External.

Цифровий вхід – soc*.

Цей вивід присутній, якщо в конфігурації встановлений прапорець "Use signal on soc terminal". Компоненти PSoC Creator можна зупиняти та запускати з допомогою функцій API вбудованого програмного забезпечення. Для забезпечення стабілізації кола перший передній фронт повинен генеруватися не раніше, ніж через 10 мкс після запуску компонента АЦП.

*Вхід тактових імпульсів** - aclk.

Цей вивід дозволяє підключати тактові імпульси PSoC до компоненти. Цей режим використовується тоді, коли важливо, щоб тактові імпульси, що використовуються АЦП, були ідентичними імпульсам синхронізації, що використовуються іншими компонентами схеми. Цей вивід можна відобразити, якщо встановити прапорець "Show analog clock (aclk) terminal". Компонент буде автоматично вибирати тактову частоту АЦП, що дозволить більш точно узгодити її з заданою користувачем частотою дискретизації.

Цифровий вихід - sdone.

Цей сигнал підвищується на протязі двох тактових імпульсів АЦП, щоб вказати, що АЦП вибрав поточний вхідний канал. Всередині цей сигнал використовується для переходу мультиплексора сигналів на наступний канал.

Цифровий вихід – eos.

Передній фронт в кінці вимірювання (eos) означає, що поточне вимірювання є закінченим. На цей момент регістри результату перетворення містять дійсні дані вибірки для всіх включених каналів. Це використовується для забезпечення переривання.

Параметри компонента SAR ADC.

Розглянемо параметри, які можна змінювати або перевіряти з допомогою налаштування параметрів компонента, згрупованих у вигляді ряду вкладок. При цьому підтримується до чотирьох конфігурацій, які вибираються під час виконання, кожна зі своїм власним умовним позначенням і додатковою вкладкою конфігурації. Для вивчення налаштування параметрів потрібно перетягнути SAR ADC на свій проект та двічі клацнути на символі компонента. При цьому відкриється діалогове вікно налаштування. Для будь-якого вибраного параметра, опція зображена тут жирним шрифтом, є значенням за замовчуванням.

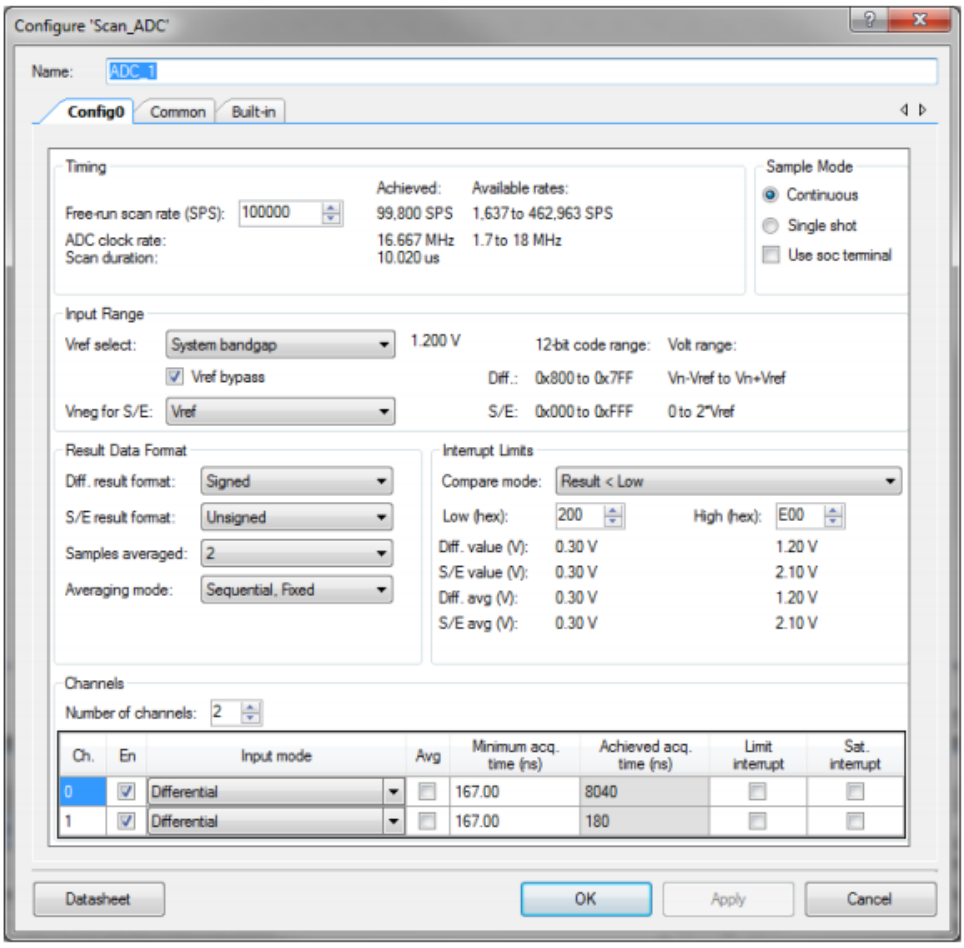

Табл. 10.1. Підвкладка – Scan вкладки Config

Синхронізація.

Частота дискретизації в вільному режимі (SPS).

Це основний параметр для Scanning SAR ADC, бажана частота дискретизації, з якою повинні виконуватися закінчення сканування, коли компонент працює в неперервному режимі. Це частота дискретизації кожного

сигналу, включеного для сканування. В налаштовувачі компонента Scanning SAR ADC є калькулятор режимів, який дозволяє наблизити цю частоту дискретизації до введеного значення. Це досягається шляхом інтелектуального вибору тактової частоти АЦП (при виборі внутрішнього тактового джерела) і часу вибірки каналу з врахуванням всіх інших заданих користувачем вимог. При виборі тактова частота АЦП автоматично розраховується на основі кількості каналів, усереднення, роздільної здатності і отримання параметрів часу для відповідності заданій частоті дискретизації.

Досягнуто (тільки відображається, не можна налаштовувати).

В цьому полі відображається досягнута в даний момент частота дискретизації, яку компонент буде реалізувати в працюючому проекті. Планувальник налаштовує все доступне, щоб максимально наблизитися до бажаної частоти дискретизації, але не завжди можна досягнути бажаної частоти дискретизації. Досягнута частота дискретизації залежить від наступного:

- АЦП;
- Кількості каналів;
- Усереднення;
- Роздільної здатності;
- Досягнутого часу перетворення.

Час вибірки для каналу – це час, необхідний для отримання аналогового сигналу і перетворення його в цифровий код.

$$Ch(i)\ Sample\ Time = Ch(i)\ Achieved\ Acquisition\ Time + \frac{(Resolution + 3)}{ADC\ Clock\ Rate}$$

Канали, використовуючи один з режимів послідовного усереднення, вибираються декілька раз в кожному скануванні. Канали, які не усереднюються а бо використовують режим усереднення з чередуванням, вибираються тільки один раз за сканування. Нехай N буде кількістю каналів в скануванні, а $Ch(i)$ $SamplesPerScan$ буде кількістю вибірок, усереднених за одне сканування для каналу. Таким чином, досягається швидкість сканування:

$$Achieved\ Scan\ Time = \sum_{i=0}^{N-1} (Ch(i)Sample\ Time) \cdot (Ch(i)SamplesPerScan)$$

$$Achieved\ Scan\ Rate = \frac{1}{Achieved\ Scan\ Time}$$

Приклад №1 конфігурації.

- Частота дискретизації АЦП – 18 МГц.
- Число каналів – 1.
- Роздільна здатність – 12 біт.
- Канал 0 (CH0):
  -- усереднення відсутнє;
  -- роздільна здатність – 12 біт;
  -- досягнутий час перетворення – 167 ns.

Досягнута частота дискретизації:

$$\text{Achieved Scan Rate} = 1/\left(\left(167\ ns + \frac{(12+3)}{18\ MHz}\right) * 1\right) = 1\ MSPS$$

Приклад №2 конфігурації.

- Частота дискретизації АЦП – 18 МГц.
- Число каналів –3.
- Роздільна здатність – 12 біт.
- Канал 0 (CH0):
  -- усереднення відсутнє;
  -- досягнутий час перетворення – 167 ns.
- Канал 1 (CH1):
  -- послідовне усереднення; сума з усередненням по 4-м вибіркам;
  -- досягнутий час перетворення – 167 ns.
- Канал 2 (CH2):
  -- усереднення відсутнє;
  -- досягнутий час перетворення – 167 ns.

Досягнута частота дискретизації:

$$1/\left(\left(\left(167\ ns\ +\ \frac{(12+3)}{18\ MHz}\right)*1\right)+\left(\left(167\ ns\ +\ \frac{(12+3)}{18\ MHz}\right)*4\right)+\left(\left(167\ ns\ +\ \frac{(12+3)}{18\ MHz}\right)*1\right)\right)=167\ kSPS$$

Доступні частоти (тільки відображення).

В цьому полі відображається приблизний мінімальний та максимальний діапазон швидкостей сканування, які зараз можуть бути досягнуті при заданому налаштуванні. Якщо бажана швидкість автономного режиму менша мінімальної швидкості, відображеної тут, одним з розв'язків є встановлення таймера TC/PWM на схемі і використання його для періодичного запуску АЦП в режимі одиничного запуску. Інші варіанти – використання усереднення або налаштування фіктивного каналу.

Частота дискретизації АЦП (тільки відображення).

В цьому полі відображається поточна вибрана фактична тактова частота АЦП. Це цілочисельне ділення PeriClk PSoC 6. Ця тактова частота налаштовується під час виконання. Це значення не обов'язково буде відповідати значенню на вкладці частоти DWR під час компіляції, якщо частота PeriClk змінюється під час виконання.

Тривалість сканування (тільки відображення).

В цьому полі задається тривалість загального сканування, у нс.

Режими дискретизації.

Скануючий SAR ADC може працювати в одному з двох режимів:

Continuous **(**неперервний**)** – після запуску сканування SAR ADC працює неперервно до зупинки.

Single shot – (одиночне сканування) – скануючий SAR ADC виконує одне сканування для кожного апаратного запуску.

Використання soc терміналу.

Скануючий SAR ADC завжди можна запустити та зупинити в прошивці (вбудованому програмному забезпеченні) з допомогою функцій ADC_StartConvert() та ADC_StopConvert().

Якщо цей прапорець встановлений, то на компоненті включений апаратний запуск через термінал початку перетворення (soc). Термінал soc створюється на символі компонента через перевірку "Use signal on soc terminal" на додатковій вкладці Scan. Якщо цей апаратний запуск включений, в режимі

одиночного сканування одне повне сканування SAR ADC ініціюється додатним фронтом імпульсу, прикладеному до терміналу soc. В неперервному режимі роботи АЦП виконує повторне сканування, якщо до терміналу прикладений рівень логічної ′1′.

Включення апаратного запуску не подавлює функцію запуску в прошивці. Потрібно обережно інтерпретувати набори даних, отриманих в результаті поєднання двох форм запуску, так як джерело запуску не відображається у вихідних даних.

Вхідний діапазон. Вибір опорної напруги Vref.

Параметр Vref вибирає джерело опорної напруги, яке використовується для ядра АЦП, і, можливо, дозволяє йому надати числове значення, якщо користувач не знає цього.

System Band-gap (системний діапазон) – виділене внутрішнє підключення до основної напруги 1.2 В.

External device pin (зовнішній вивід пристрою) – деякі пристрої PSoC 6 підтримують виділений вивід Vref, що використовується для байпасного (обхідного) конденсатора поза кристалом і для вводу опори, зовнішньої по відношенню до кристалу. Прапорець обходу Vref не діє в цьому режимі. Якщо вибраний пристрій PSoC 6 не підтримує цей виділений вивід, то опорний параметр не буде відображатися.

Vdda/2 – внутрішній резистивний дільник видає Vdda/2 як опорну напругу.

Vdda – використовується внутрішня аналогова напруга живлення, що подається на виводи Vdda. Обхідний конденсатор поза кристалом не працює в цьому режимі.

Величина Vref (вводиться користувачем або відображення параметрів).

Справа від Vref виберіть pull-down. Цей параметр або відображає значення опорної напруги, яке використовується для SAR ADC або дозволяє ввід значення з метою відображення, якщо тільки користувач знає це значення. Величина напруги Vref повинна бути не меншою 0,85 В, і його задання викличе помилку.

Обхід Vref.

Встановлення цього прапорця вказує налаштовувачу компонент, що підключено обхідний конденсатор поза кристалом до спеціального виводу пристрою, призначеному для цього. Це дозволяє компоненті вибирати більш високі тактові частоти АЦП і, отже, значно більш високі загальні частоти сканування.

Рекомендується використовувати еталонний байпасний конденсатор поза кристалом (50 нФ або більше) у всіх системах.

Якщо вибраний пристрій PSoC 6 не підтримує виділений вивід пристрою для обхідного конденсатора поза кристалом, цей прапорець не буде відображатися.

Vneg для S/E.

Цей параметр вибирає, де підключений від’ємний вхід до SAR ADC, якщо канали налаштовані для односторонньої роботи.

Vssa – Діапазон вхідних сигналів від 0,0 до Vref, ефективна роздільна здатність буде на один біт меншою, ніж вибрана в налаштувальнику.

Vref - Діапазон вхідних сигналів від 0,0 до Vref*2.

External – Цей режим налаштований для квазі-диференційних входів. Декілька каналів спільно використовують одне спільне ve (інвертоване) зєднання. Це часто використовується для подавлення синфазного шуму "землі" в багатоканальних системах.

12-ти бітний діапазон коду (*тільки відображення*).

В цьому полі відображається діапазон кодів, який буде повернутий SAR ADC. Значення, що відображаються, усікаються до 12 бітів. Але результати, що повертаються, можуть бути розширені до 16 або 32-х бітного формату, в залежності від того, яка функція GetResult використовується.

Діапазон напруг (*тільки відображення*).

В цьому полі відображається діапазон напруг SAR ADC з використанням вибраної опорної напруги Vref. Для несиметричних каналів вибір Vneg також використовується для визначення діапазону.

Формат даних результату.

Формат диференційного (*Diff.*) *результату*.

Цей параметр визначає, чи є результат диференційного вимірювання знаковим або без знаковим. Це глобальне налаштування для всіх диференційних каналів. Результати завжди обґрунтовано правильні.

S/E формат результату.

Цей параметр визначає, чи є результат одностороннього вимірювання знаковим або без знаковим. Це глобальне налаштування для всіх несиметричних каналів. Результати завжди обґрунтовано правильні.

В табл. 10.1 показано, як ці параметри впливають на перетворення вхідної напруги в значення 12-ти розрядної цифрової вибірки.

Табл. 10.1.

s/e або diff	Signed / Unsigned	Single-ended negative input	-Input	+Input	Result Register
s/e	Unsigned: (обережно використ. цей режим)	Vssa	Vssa	Vref Vssa -noise	0x0FFF 0x0800 0x07xx
s/e	Signed	Vssa	Vssa	Vref Vssa -noise	0x07FF 0x0000 0xFFxx
s/e	Signed	External	Vneg	Vx+Vref Vx Vx-Vref	0x07FF 0x0000 0xF800
s/e	**Unsigned**	**Vref**	**Vref**	**2*Vref** **Vref** **Vssa**	**0x0FFF** **0x0800** **0x0000**
s/e	Signed	Vref	Vref	2*Vref Vref Vssa	0x07FF 0x0000 0xF800
diff	Unsigned	N/A	Vx	Vx+Vref Vx Vx-Vref	0x0FFF 0x0800 0x0000
diff	**Signed**	**N/A**	**Vx**	**Vx+Vref**	**0x07FF**

				Vx **Vx-Vref**	**0x0000** **0xF800**

Для односторонніх перетворень з параметром Vneg для s/e, встановленому на Vssa, корисне перетворення фактично є 11 бітним. Шум або зміщення на вході +Input з рівнем трохи нижчим Vssa приводить до результату, який виявляється більш позитивним, ніж повна шкала. Це може викликати системні проблеми. Тому цей режим потрібно використовувати обережно.

Усереднення вибірки.

Цей параметр встановлює коефіцієнт усереднення для будь-якого каналу з увімкненою опцією усереднення. Це глобальне налаштування для всіх каналів, у яких включено усереднення. Значення за замовчуванням - 2.

Режим усереднення.

Цей параметр встановлює, як працює режим апаратного усереднення. Якщо вибрано Sequential, Sum, кожний результат перетворення АЦП додається до поточної суми. Потім він зсувається в кінці сканування, щоб відповідав 16-ти бітному результуючому слову. Якщо вибрано Sequential, Fixed режим, то накопичений результат перетворюється до 12-ти бітного результату.

У будь-якому послідовному режимі сканування призупиняється на каналі, що усереднюється, і всі вибірки для середнього значення беруться перед тим, як перейти до наступного каналу сканування. Це може істотно знизити максимальну доступну швидкість сканування, коли будь-який канал при скануванні усереднюється таким чином.

З цієї причини також доступний режим Interleaved, Sum. У режимі Interleaved перед переходом виконується лише одне перетворення на кожному каналі. Але канали, в яких включено усереднення, отримують попередньо встановлену кількість вибірок, накопичених у їхньому регістрі результатів.

В режимі Interleaved, Sum загальна швидкість сканування не зменшується. Це означає, що канали, що не потребують усереднення, все ще можуть бути дискретизовані з початковою швидкістю сканування. Переривання кінця сканування все ще генеруються в кінці кожного сканування. Канали, які виконують interleaved усереднення, не помечаються як "дійсні", поки не буде виконано правильну кількість сканувань.

Якщо для кожного каналу встановлено усереднення, а для режиму встановлено значення Interleaved, Sum, швидкість перервання в кінці сканування значно знижується. Переривання відбудеться, коли всі канали отримають нові "дійсні" дані після завершення такої ж кількості сканувань, що й для параметру усереднені вибірки.

За замовчуванням PSoC Creator призначає ім'я екземпляра "ADC_1" першій компоненті в проекті. Його можна перейменувати на будь-яке унікальне значення, яке відповідає синтаксичним правилам для ідентифікаторів. Назва екземпляра стає префіксом кожного глобального імені функції, змінної та постійного символу. Для зручності ім'я екземпляра, яке використовується далі, - "ADC".

ADC_Start() - Виконує всю необхідну ініціалізацію для цього Компонента та вмикає живлення. Потужність буде встановлена на відповідну потужність на основі тактової частоти перетворення.

ADC_StartEx() - Виконує ту ж функцію, що і функція ADC_Start(), а також встановлює вектор переривання на визначену користувачем адресу.

ADC_Stop() - Функція зупиняє перетворення АЦП і переводить його в режим найнижчої потужності.

ADC_SelectConfig() - Вибирає заздалегідь задану конфігурацію для сканування.

ADC_StartConvert() - У неперервному режимі запускає процес перетворення і він працює постійно. У режимі запуску ця функція запускає кожне перетворення.

ADC_StopConvert() - Примушує АЦП припинити перетворення. Якщо в даний час здійснюється перетворення, то воно буде завершене, але подальше перетворення не відбудеться.

ADC_SetConvertMode() - Встановлює режим перетворення на одиночний або неперервний.

ADC_IRQ_Enable() - Включає переривання в кінці перетворення. Також повинні бути включені глобальні переривання, щоб відбулися переривання АЦП.

ADC_IRQ_Disable() - Вимикає переривання в кінці перетворення.

ADC_SetEosMask() - Встановлює або очищає біт маски переривання кінця сканування (EOS).

ADC_SetChanMask() - Встановлює включення / виключення маски для всіх каналів.

ADC_IsEndConversion() - Негайно повертає статус перетворення або не повертає (блокує), поки не закінчиться перетворення, залежно від параметра retMode.

ADC_GetResult16() - Отримує дані, доступні в регістрі результатів SAR, повертає 16-ти розрядні дані.

ADC_GetResult32() - Отримує дані, доступні в регістрі результатів SAR, повертає 32-х розрядні дані.

ADC_SetLowLimit() - Цей параметр встановлює нижню межу для порівняння границь.

ADC_SetHighLimit() - Цей параметр встановлює верхню межу для порівняння границь.

ADC_SetLimitMask() - Встановлює, які канали можуть спричинити переривання по граничній умові.

ADC_SetSatMask() - Встановлює, які канали можуть призвести до переривання через подію насичення.

ADC_SetOffset() - Встановлює зміщення каналу АЦП.

ADC_SetGain() - Встановлює коефіцієнт підсилення в підрахунку на 10 Вольт для каналу АЦП.

ADC_CountsTo_Volts() - Перетворює вихідний код АЦП у Вольти (формат – число з плаваючою комою).

ADC_CountsTo_mVolts() - Перетворює вихідний код АЦП в мілівольти (мВ).

ADC_CountsTo_uVolts() - Перетворює вихідний код АЦП в мікровольти (мкВ).

Підключимо до нашого проекту компонент SAR ADC, який вимірюватиме аналогову напругу на змінному резисторі (рис. 10.2).

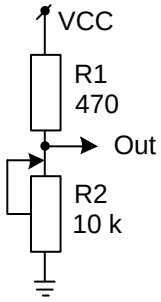

Рис. 10.2. Схема вимірювання напруги на виході Out

Підключимо вивід Out до порту P10.4 PSoC 6. Для цього потрібно налаштувати SAR ADC для читання напруги на цьому виводі.

При пошуку АЦП в Component Catalog знайдемо скануючий SAR ADC, який перетягнемо на проект верхнього рівня (рис. 10.13).

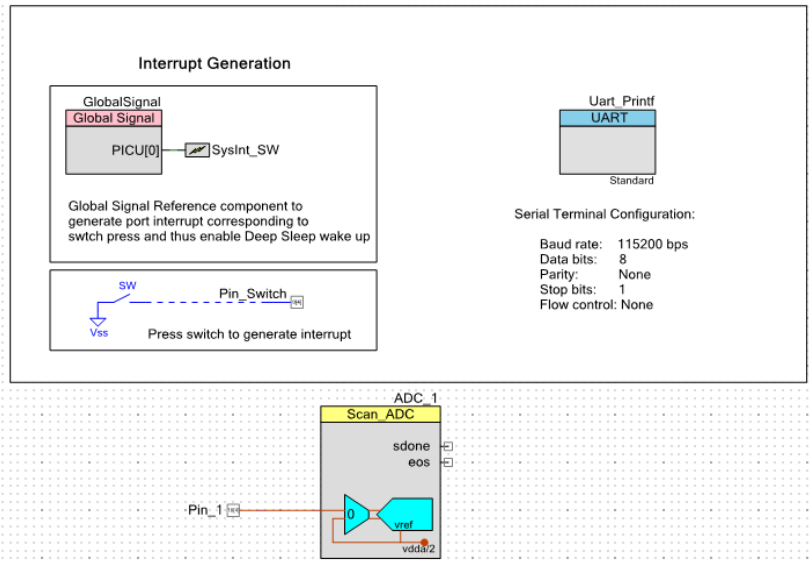

Рис. 10.3. Схема проекту з компонентою SAR ADC

Налаштуємо параметри SAR ADC так як зображено на рис. 10.4.

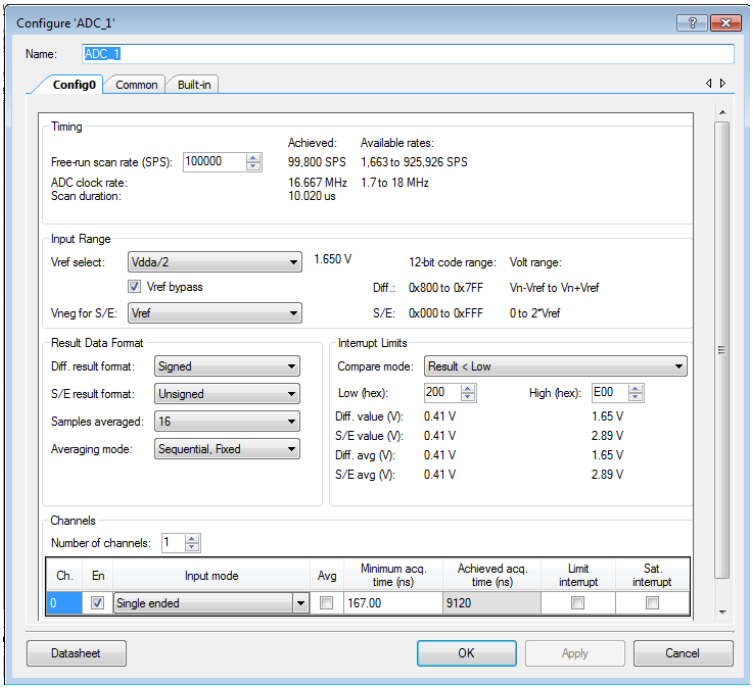

Рис. 10.4. Вигляд вкладки налаштування параметрів SAR ADC

Підключимо цей вивід до порту P10.4 (рис. 10.5).

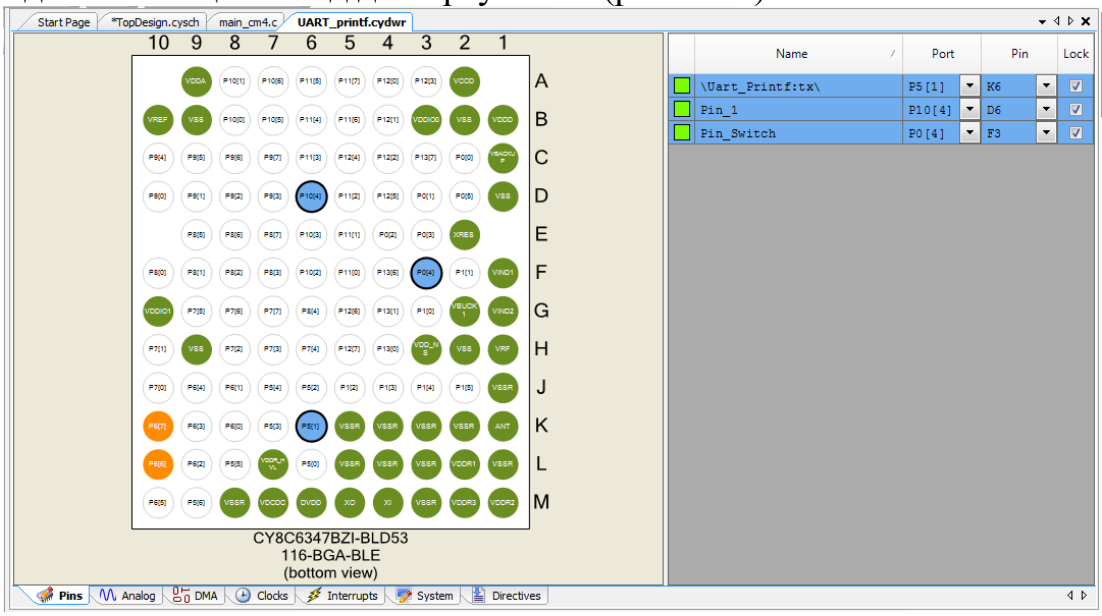

Рис. 10.5. Карта підключення виводів в проекті

Реалізація коду програми для вимірювання напруги на виході Out схеми (рис. 10.2).

```
#include "project.h"
#include <stdio.h>

#define MAX_COUNT 65535u

/*******************************************************************************
*        Function Prototypes
*******************************************************************************/
static void HandleError(void);
static void SwInterruptIsr(void);


/*******************************************************************************
```

```
*        Global variables
*******************************************************************************/
int count = 0u;
uint32_t printCount = false;

/*******************************************************************************
* Function Name: SwInterruptIsr
********************************************************************************
* Summary:
* This function is executed when GPIO interrupt is triggered.
* This ISR
*        i. Clears the interrupt source.
*        ii. Increments the count and sets printCount flag.
*
* \Param None
*
* \return
* void
*
*******************************************************************************/
static void SwInterruptIsr(void)
{
    Cy_GPIO_ClearInterrupt(Pin_Switch_0_PORT, Pin_Switch_0_NUM);

    count++;
    /* If count reaches MAX_COUNT reset the count variable. */
    if(count == MAX_COUNT)
    {
        count = 0u;
    }
    printCount = true;

      NVIC_ClearPendingIRQ(SysInt_SW_cfg.intrSrc);
}

/*******************************************************************************
* Function Name: main
********************************************************************************//**
*
* Summary:
*  The main function performs the following actions:
*   1. Sets up UART component.
*   2. If initialization of UART component fails then stay in an infinite loop.
*   3. Continuously check whether printCount flag is set.
*   4. If printCount flag is set, reset this flag and print the count using
*      printf function.
*
* Parameters:
*  None
*
* Return:
*  None
*
*******************************************************************************/
int main(void)
{
    int16_t adc_Code, adc_V;

    /* UART initialization status */
    cy_en_scb_uart_status_t uart_status ;

     /* Initialize UART operation. Config and Context structure is copied from
Generated source. */
```

```
    uart_status  = Cy_SCB_UART_Init(Uart_Printf_HW, &Uart_Printf_config,
&Uart_Printf_context);
    if(uart_status != CY_SCB_UART_SUCCESS)
    {
        HandleError();
    }
    Cy_SCB_UART_Enable(Uart_Printf_HW);
    ADC_1_Init();
    ADC_1_Start();
    ADC_1_StartConvert();

    /* Initialize and enable GPIO interrupt assigned to CM4+ */
    /* SysInt_SW_cfg structure is defined in cyfitter_sysint_cfg.c based on
       parameters entered in the Interrupts tab of CYDWR. */
    Cy_SysInt_Init(&SysInt_SW_cfg, SwInterruptIsr);
    NVIC_ClearPendingIRQ(SysInt_SW_cfg.intrSrc);
    NVIC_EnableIRQ((IRQn_Type)SysInt_SW_cfg.intrSrc);

     __enable_irq(); /* Enable global interrupts. */

    printf("\033[2J\033[H"); // Очищення екрану
    printf(" -- Lab_10_1. --\r\n");
    printf(" -- Work with SAR ADC in PSoC 6. --\r\n");
    for(;;)
    {
        if(printCount)
        {
            printCount = false;
            printf("Number of times switch SW2 has been pressed = %d\r\n",
count);
        }
        ADC_1_IsEndConversion(1);
        adc_Code = ADC_1_GetResult16(0);
        adc_V = ADC_1_CountsTo_Volts(0,adc_Code);
        printf("  --- ADC Code = %d --- ADC Value = %d (V) --- \r\n", adc_Code,
adc_V);
        CyDelay(100);
    }
}

/*******************************************************************************
* Function Name: HandleError
********************************************************************************
* Summary:
* This function processes unrecoverable errors such as UART component
* initialization error. In case of such error the system will stay in an
* infinite loop of this function.
*
* Parameters:
*  None
*
* Return:
*  None
*
*******************************************************************************/
static void HandleError(void)
{
     /* Disable all interrupts */
    __disable_irq();

    while(1u)
    {

    }
```

```
}


/*******************************************************************************
* When printf function is called it is redirected to the following functions
* depending on compiler used.
*******************************************************************************/
#if defined (__GNUC__)
/*******************************************************************************
* Function Name: _write
********************************************************************************
* Summary:
* NewLib C library is used to retarget printf to _write. printf is redirected to
* this function when GCC compiler is used to print data to terminal using UART.
*
* \param file
* This variable is not used.
* \param *ptr
* Pointer to the data which will be transfered through UART.
* \param len
* Length of the data to be transfered through UART.
*
* \return
* returns the number of characters transferred using UART.
* \ref int
*******************************************************************************/
   int _write(int file, char *ptr, int len)
    {
        int nChars = 0;

        /* Suppress the compiler warning about an unused variable. */
        if (0 != file)
        {
        }

        nChars = Cy_SCB_UART_PutArray(Uart_Printf_HW, ptr, len);

        return (nChars);
    }
#elif defined(__ARMCC_VERSION)

/*******************************************************************************
* Function Name: fputc
********************************************************************************
* Summary:
* printf is redirected to this function when MDK compiler is used to print data
* to terminal using UART.
*
* \param ch
* Character to be printed through UART.
*
* \param *file
* pointer to a FILE object that identifies the stream where the character is to
be
* written.
*
* \return
* This function returns the character that is written in case of successful
* write operation else in case of error EOF is returned.
* \ref int
*******************************************************************************/
    struct __FILE
    {
        int handle;
```

```
};

enum
{
    STDIN_HANDLE,
    STDOUT_HANDLE,
    STDERR_HANDLE
};

FILE __stdin = {STDIN_HANDLE};
FILE __stdout = {STDOUT_HANDLE};
FILE __stderr = {STDERR_HANDLE};

int fputc(int ch, FILE *file)
{
    int ret = EOF;
    switch(file->handle)
    {
        case STDOUT_HANDLE:
            while (0UL == Cy_SCB_UART_Put(Uart_Printf_HW, ch))
            {
            }
            ret = ch;
        break;

        case STDERR_HANDLE:
            ret = ch;
        break;

        default:
            file = file;
        break;
    }
    return(ret);
}


#endif /* (__GNUC__) */


/* [] END OF FILE */
```

Завдання.

1. Реалізувати вимірювання опору та вивести результати на термінал.

2. Реалізувати вимірювання температури з допомогою термістора.